Computing Straight Skeletons and Motorcycle Graphs: Theory and Practice

Stefan Huber

June 2011

Submitted to the Faculty of Natural Sciences at the
University of Salzburg in partial fulfillment of the
requirements for the Doctoral degree.

Supervisor: Ao.Univ.Prof. DI Dr. Martin Held
Department of Computer Science
University of Salzburg, Austria

ABSTRACT

The straight skeleton is a geometric structure that is similar to generalized Voronoi diagrams. Straight skeletons were introduced to the field of computational geometry one and a half decades ago. Since then many industrial and academical applications emerged, such as the computation of mitered offset curves, automatic roof construction, solving fold-and-cut problems and the reconstruction of surfaces, to name the most prominent ones. However, there is a significant gap between the most efficient straight-skeleton algorithms and implementations, on one hand, and the best known lower runtime bounds on the other hand. The primary goal of this thesis is the development of an algorithm that is suitable for implementation and efficient in terms of time and space in order to make the advantages of straight skeletons available to real-world applications.

We start with investigations concerning upper and lower bounds on the number of so-called flip events that occur in the triangulation-based straight-skeleton algorithm by Aichholzer and Aurenhammer. In particular, we prove the existence of Steiner triangulations that are free of flip events. This result motivates a novel straight-skeleton algorithm for non-degenerate simple polygons that is based on the so-called motorcycle graph. In order to extend this algorithm to arbitrary planar straight-line graphs, we carefully generalize the motorcycle graph. This generalization leads to practical and theoretical applications: Firstly, we obtain an extension of the alternative characterization of straight skeletons by Cheng and Vigneron to planar straight-line graphs. Secondly, this characterization motivates a straight-skeleton algorithm that is based on 3D graphics hardware. Thirdly, the generalized motorcycle graph leads to a wavefront-type straight-skeleton algorithm for arbitrary planar straight-line graphs. Our algorithm is easy to implement, has a theoretical worst-case time complexity of $O(n^2 \log n)$ and operates in $O(n)$ space. Extensive runtime tests with our implementation Bone exhibit an actual runtime of $O(n \log n)$ on a database containing more than 13 500 datasets of different characteristics. In practice, this constitutes an improvement of a linear factor in time and space compared to the current state-of-the-art straight-skeleton code, which is shipped with the CGAL library. In particular, Bone performs up to 100 times faster than the current CGAL code on datasets with a few thousand vertices, requires significant less memory and accepts more general input.

Our straight-skeleton algorithm motivates the investigation of motorcycle graphs and their practical computation. We start with stochastic considerations of the average length of a motorcycles trace. The results obtained motivate a simple yet fast algorithm that employs geometric hashing. Runtime tests with our implementation Moca exhibit an $O(n \log n)$ runtime on the vast majority of our datasets. Finally, we revisit the geometric relation of straight skeletons and motorcycle graph. We present an algorithm that constructs planar straight-line graphs whose straight skeleton approximates any given motorcycle graph to an arbitrary precision. This algorithm finally leads to a P-completeness proof for straight skeletons of polygons with holes that is based on the P-completeness of motorcycle graphs.

ACKNOWLEDGMENTS

I would like to thank my parents, Stefan and Gabriele, for their unlimited and unconditional support throughout my entire life. Special thanks go to my girlfriend Christina for her constant support and understanding during my doctoral studies.

I also thank my supervisor Martin Held for his guidance and his extensive support. Further, I thank my colleagues and co-workers. In particular I thank Gerhard Mitterlechner and Roland Kwitt for their proofreading and valuable discussions.

Eventually, I thank the Austrian Science Fund (FWF) for supporting this thesis by funding project no. L367-N15.

CONTENTS

1 INTRODUCTION

Assume we are given a simple[1] polygon in the plane which constitutes the outer walls of a house in a ground plan. How can we automatically build a roof such that all faces of the roof have equal slope and no rain drop gets caught in a sink? Assume we are given a polygon with holes[2] which is to be milled out with a numerically controlled machine (NC machine). How do we compute so-called offset curves of the input such that sharp vertices of the input remain sharp for the offset curve?[3] Assume we are given a simple polygon drawn on a paper. How can we determine a series of complete folds of the paper and a subsequent single cut through the folded paper such that one of the resulting paper pieces has the shape of the polygon? These three problems share a common feature: they can be solved using the *straight skeleton*.

Roughly speaking, the straight skeleton of a simple polygon P is a tree-like skeleton structure that is similar to Voronoi diagrams of polygons, but consists of straight-line segments only. The definition of the straight skeleton is based on the so-called wavefront propagation process. In a nutshell, the straight skeleton of P consists of the set of points that are traced out by the vertices of P when the edges of P shrink inwards in a self-parallel manner and with constant speed. During this propagation process, some edges may vanish and the polygon may successively disintegrate into multiple parts, see Figure 1. Straight skeletons of simple polygons were introduced by Aichholzer et al. [AAAG95] and were subsequently generalized to planar straight-line graphs[4] by Aichholzer and Aurenhammer [AA96] about one and a half decades ago. Similar to Voronoi diagrams, straight skeletons possess a heterogeneous plenitude of applications and many of them appeared just in the past decade. The list of applications includes

- the computation of mitered offset curves and motion planning in computer-aided design and computer-aided manufacturing (CAD/CAM) [PC03];
- computing the quickest walking paths in a Manhattan-style city with a fast public transit by means of the city Voronoi diagram; computing critical areas in VLSI circuit models [AAP04, Pap98];
- automatic roof generation, terrain modeling and area collapsing within maps in geographical information systems (GIS) [AAAG95, AA96, LD03, MWH+06, KW11, Hav05, HS08];

1 A polygon is called simple if it is not self-overlapping. That is, each vertex is incident to exactly two segments and two segments may only intersect at their endpoints.

2 A polygon with holes is a simple polygon P, from which a finite number of simple polygons is removed.

3 A robust, easy and well-known way to compute offset curves is based on Voronoi diagrams. However, Voronoi-based offset curves contain circular arcs around vertices that point inside the polygon.

4 A planar straight-line graph consists of a finite set of vertices in the plane which are connected by straight-line edges such that two edges may only intersect in their endpoints.

- solving fold-and-cut problems, computing hinged dissections of polyhedra and related problems in the field of mathematical origami [DO07, DDL98, DDM00, DDLS05];
- shape reconstruction and interpolation of contour lines in three-dimensional space [OPC96, BGLSS04];
- and polygon decompositions [TV04b].

Some of these applications are genuine to straight skeletons, like fold-and-cut problems or automatic roof construction. Other applications are inherited from generalizations of Voronoi diagrams, like many problems in the fields of CAD/CAM, VLSI and GIS.

In contrast to the large number of applications, we perceive a gap between available algorithms and implementations on the one hand and the best known lower time bound on the other hand. At the moment, the best known lower time bound for the computation of straight skeletons is $\Omega(n)$ for an n-vertex polygon and $\Omega(n \log n)$ for an n-vertex planar straight line graph. However, the currently fastest algorithm for arbitrary polygons and planar straight-line graphs is due to Eppstein and Erickson [EE99] and has a theoretical worst-case time complexity of $O(n^{17/11+\epsilon})$, where $\epsilon > 0$ is an arbitrary small number. The only straight-skeleton implementation available is shipped with the CGAL library [CGA] and was implemented by Cacciola [Cac04]. However, our experiments revealed that the underlying algorithm needs $O(n^2 \log n)$ time and $O(n^2)$ space for practical input data.

This thesis deals with theoretical properties and the practical computation of straight skeletons and motorcycle graphs. The goal is to develop a practice-minded straight-skeleton algorithm, which accepts arbitrary planar straight-line graphs as input, is efficient in time and space and easy to implement.

1.1 ORGANIZATION

In this chapter, we start with the definition of straight skeletons, the terrain model and the motorcycle graph in Section 1.2. We continue with a presentation of several applications of straight skeletons in Section 1.3. In Section 1.4, we review known algorithms and implementations of straight skeletons and motorcycle graphs and in Section 1.5, we discuss different approaches by which straight skeletons were generalized.

Chapter 2 is devoted to the computation of straight skeletons. In Section 2.1, we collect known geometric properties of the straight skeleton and present them along with their proofs in a unified formalism based on the definitions in Section 1.2. In Section 2.2, we analyze the triangulation-based algorithm by Aichholzer and Aurenhammer [AA98]. We prove a few results concerning the lower and upper bounds for the number of the so-called flip events and show that Steiner triangulations exist where no flip events occur.

This insight motivates a novel straight-skeleton algorithm for simple non-degenerate[5] polygons that is based on the motorcycle graph in Section 2.3. In order to extend this approach to arbitrary planar straight-line graphs, we carefully generalize the motorcycle graph in Section 2.4. Further, we prove two essential geometric properties for this generalization that are important for our algorithmic approach: (i) the input graph and the motorcycle

5 See Section 1.2.4 for the definition of the non-degeneracy assumption by Cheng and Vigneron.

graph tessellate the plane into convex faces and (ii) the generalized motorcycle graph covers the reflex arcs of the straight skeleton.

In addition, the generalized motorcycle graph permits an extension of Cheng and Vigneron's alternative characterization of straight skeletons to arbitrary planar straight-line graphs. Furthermore, this characterization motivates a straight-skeleton algorithm that employs 3D graphics hardware to approximately compute the straight skeleton.

In Section 2.5, we present a wavefront-type straight-skeleton algorithm for arbitrary planar straight-line graphs. Our implementation Bone runs in $O(n^2 \log n)$ time in the worst-case and uses $O(n)$ space. Experiments exhibit an actual $O(n \log n)$ runtime on real-world input. This constitutes a speed-up by a linear factor in time and space compared to the current state-of-the-art straight-skeleton code that is shipped with the CGAL library.

In Chapter 3 we have a closer look on the motorcycle graph. Our straight-skeleton implementation Bone requires a fast motorcycle graph implementation for practical input. We start our investigations with stochastic considerations of the average trace length in a motorcycle graph in Section 3.2. Motivated by the results we obtained, we present a simple motorcycle graph algorithm, which uses geometric hashing to speed up the computation, see Section 3.3. Runtime tests show an actual runtime of $O(n \log n)$ on practical input for our implementation Moca.

In Section 3.4, we further investigate the geometric relation between the straight skeleton and the motorcycle graph. We first show how to construct a planar straight-line graph whose straight skeleton approximates the motorcycle graph up to any given precision. Further, we present a simple algorithm that computes the motorcycle graph using the straight skeleton. This algorithm finally leads to a LOGSPACE-reduction of the motorcycle graph problem to the straight skeleton problem, which proves the P-completeness of straight skeletons of polygons with holes based on the P-completeness of the motorcycle graph.

1.2 PRELIMINARIES AND DEFINITIONS

1.2.1 The straight skeleton of a simple polygon

Aichholzer et al. [AAAG95] introduced the straight skeleton of simple polygons P by considering a so-called *wavefront-propagation process*. Each edge e of P sends out a wavefront which moves inwards at unit speed and is parallel to e. The wavefront of P can be thought to shrink in a self-parallel manner such that sharp corners at reflex[6] vertices of P remain sharp, see Figure 1. During this propagation process topological changes, so-called *events*, will occur: Edges collapse to zero length and the wavefront may be split into multiple parts. Each wavefront vertex moves along the bisector of two edges of P and, while it moves, it traces out a straight-line segment.

Definition 1.1 (straight skeleton, arcs, nodes, faces)**.** The *straight skeleton* $\mathcal{S}(P)$ of the polygon P comprises the straight-line segments that are traced out by the wavefront vertices.

6 A vertex v of a polygon is called reflex if the angle at v at the polygon side is at least 180°. In computational geometry one rather speaks of "reflex" vertices than of "concave" vertices.

These straight-line segments and are called the *arcs* of $\mathcal{S}(P)$. The loci of the topological changes of the wavefront are called *nodes*. To each edge e of P belongs a *face* $f(e)$, which consists of all points that are swept by the wavefront edge that is sent out by e.

We illustrate the straight skeleton $\mathcal{S}(P)$ of a simple polygon P in Figure 1. Each node is incident to multiple arcs. The boundary of a face consists of arcs and the vertices of a face are nodes.

Lemma 1.2 ([AAAG95])**.** *The straight skeleton $\mathcal{S}(P)$ of a simple polygon P with n vertices is a tree. It consists $n-2$ nodes and $2n-3$ arcs and tessellates P into n faces.*

For the proof of this lemma we create a single node for each topological change, even if the resulting nodes coincide geometrically. In particular, we assume that each node has degree three.[7] For example, if P is a regular n-gon then $\mathcal{S}(P)$ has a star form, i. e. all arcs meet in the center point of P and multiple nodes with degree three are geometrically coincident.

Proof. The number of faces is n, since P comprises n edges. Next, we note that the face $f(e)$ of an edge e is connected because it is given by the set of points which are swept by a continuously moving wavefront edge. On the other hand, each point within P is reached by the wavefront after some time. Hence, P is tessellated by the faces of $\mathcal{S}(P)$ and $\mathcal{S}(P)$ consists of the boundaries of the faces. From that it follows that $\mathcal{S}(P)$ does not contain a cycle and is therefore a tree. The inner nodes of this tree are of degree 3 and the tree has n leaves. It follows that $\mathcal{S}(P)$ has $n-2$ nodes and $2n-3$ arcs. □

Definition 1.3 (reflex/convex wavefront vertex)**.** A wavefront vertex v is *reflex* if the angle on the side where v propagates to is at least $180°$. Likewise, we define a *convex* wavefront vertex.

Let us discuss the different types of topological events that occur for the wavefront in more detail. We follow Aichholzer et al. [AAAG95], who distinguish two types of events, namely *edge events* and *split events*.

- **Edge event**: An edge event occurs when two neighboring convex vertices u and v of the wavefront meet. This event causes the wavefront edge e, which connects u and v, to collapse to zero length. The wavefront edge e is removed and the vertices u and v are merged into a new convex vertex with its own velocity, see Figure 2 (a).
- **Split event**: A split event occurs when a reflex vertex u of the wavefront meets an edge e of the wavefront. The vertex u splits the entire wavefront into polygonal parts. Each part keeps on propagating for its own, see Figure 2 (b).

  Split events can occur in interesting variations. In Figure 2 (c), a split event for the reflex vertex u and the edge e occurs, while at the same time the edge between u and v collapses to zero length. Note that in this case only one wavefront part remains after the split event. However, the question arises whether we would like to call this event an edge event as well. For subfigure (a) we observe that a local disk around the

7 Strictly speaking, we also have to assume that multi split events — i. e. multiple reflex arcs meeting in a point, see below — do not occur either. If we take multi split events into account then we get at most $n-2$ nodes and at most $2n-3$ arcs.

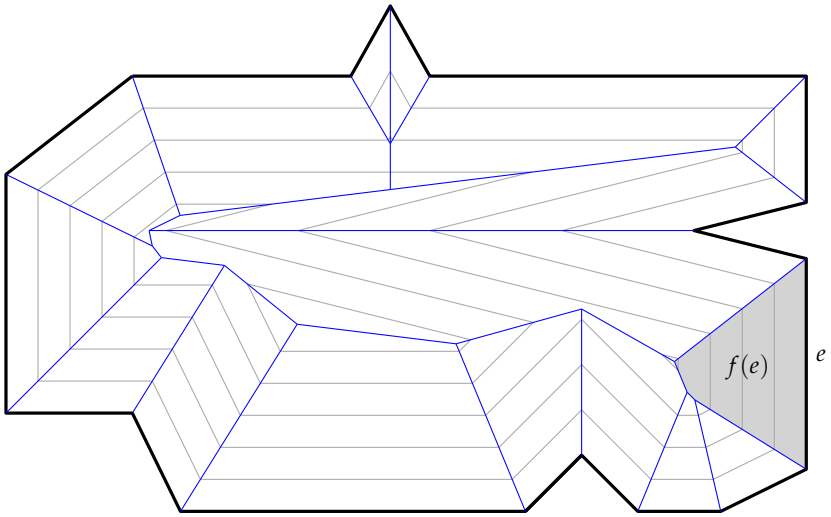


Figure 1: The straight skeleton $\mathcal{S}(P)$ (blue) of a simple polygon P (bold) is defined by a wavefront propagation process where the edges of P move inwards in a self-parallel manner. Five wavefronts at equally spaced points in time are shown in gray. The blue straight-line segments are called the arcs of $\mathcal{S}(P)$ and the common endpoints of arcs are called the nodes of $\mathcal{S}(P)$. We shade the face $f(e)$ of one edge e in light gray.

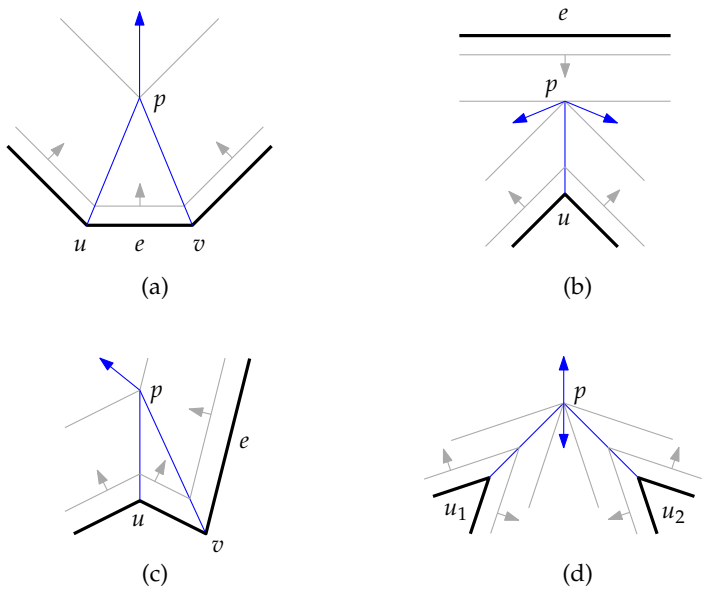


Figure 2: Two types of events occur during the wavefront propagation: edge and split events. (a) illustrates an edge event, (b) a simple split event, (c) a split event with an edge collapse combined, (d) a split event with two reflex vertices u_1 and u_2 involved. The arcs are depicted in blue and the wavefronts in gray.

point p, where the event happened, is tessellated into convex slices by the arcs of the straight skeleton. It is easy to see that this holds in general if u and v are convex. On the other hand, in subfigures (b) and (c), a tessellation of a local disk around p also contains a reflex slice. We would like to use this property as an indicator that a split event happened at p. For this reason, we call the event illustrated by subfigure (c) a split event.[8] As a consequence, edge events describe edge collapses between two convex vertices. Furthermore, if the straight-skeleton face of an edge — interpreted as a polygon — contains a reflex vertex, it has been generated by a split event.

In subfigures (b) and (c) only one reflex vertex u was involved in the split event. However, it could happen that two or more reflex wavefront vertices $u_1, \ldots, u_k$ meet each other in a common point p at the same time, see Figure 2 (d). Such situations occur easily for rectilinear[9] polygons. As a consequence the wavefront is split into k parts. Note that it is possible that a new reflex vertex emerges, as illustrated in Figure 2 (d). We will call events, where two or more reflex vertices meet simultaneously in a point a *multi split event*. Multi split events that cause a new reflex vertex to emerge are called *vertex events*.[10]

We want to remark that the taxonomy of the different classes of events is in flux within the literature of straight skeletons. For example, Tănase [TA09] pursues a different approach by considering three sub-classes of edge events and three sub-classes of split events. Following this taxonomy, the situation in Figure 2 (c) would be called *reflex edge annihilation*.

Using the taxonomy we presented above, the propagation of a wavefront always proceeds as follows: Every edge event reduces the number of wavefront edges by one. A split event splits the wavefront into polygonal parts and implies a hierarchy of nested polygons. Further, a split event reduces the number of reflex vertices at least by one. Even if a vertex event happens, at least two reflex vertices must have been involved in this event. Eventually, all nested parts in the hierarchy become convex polygons. The convex polygons are reduced by a series of edge events. Finally, a polygon vanishes by collapsing either to a point or to a straight-line segment. As an example for the first case one could imagine P to be a triangle, and for the second case a non-equilateral rectangle.

1.2.2 The straight skeleton of a planar straight-line graph

Aichholzer and Aurenhammer [AA96] generalized the concept of straight skeletons to planar straight-line graphs.[11] Assume we are given a planar straight-line graph G with no isolated[12] vertices. The basic idea to define the straight skeleton $\mathcal{S}(G)$ of G remains the same: Every edge e of G sends out two wavefront edges — one at each side of e — which travel at unit speed and stay parallel to the original edge e. However, the terminal[13] vertices of G play an interesting role, since it is a-priori unclear how the wavefront is expected to behave

8 Note that Eppstein and Erickson [EE99] call this event an edge event. Vyatkina [Vya09b] introduced the new name *sticking event*. Aichholzer et al. [AAAG95] did not discuss this case.
9 A polygon is rectilinear if its edges are axis-parallel.
10 This term was introduced by Eppstein and Erickson [EE99].
11 A later version of this paper was published in [AA98].
12 A vertex is called isolated if there are no edges incident to the vertex.
13 A terminal vertex, or "terminal" for short, is a vertex for which the number of incident edges of 1.

at such vertices. Aichholzer and Aurenhammer [AA96] decided that each terminal vertex v sends out a wavefront for it own, which is perpendicular to the single incident edge of v. As a result, the wavefront in the neighborhood of a terminal vertex v forms a rectangular cap around v, see Figure 3.

The terminology of *arcs* and *nodes* remains the same in this general setting. We still have the two types of events — *edge events* and *split events* — which occur during the propagation of the wavefronts and we define $\mathcal{S}(G)$ as the set of points that are traced out by the vertices of the wavefront. Note that a vertex v of G with degree $k \geq 2$ causes k wavefront vertices to emanate from v. Furthermore, every terminal vertex v causes two reflex wavefront vertices to emanate, see Figure 3.

Definition 1.4 (wavefront)**.** We denote by $\mathcal{W}(G,t)$ the *wavefront* of a planar straight-line graph G at some time $t \geq 0$.

We interpret $\mathcal{W}(G,t)$ for a fixed t as a 2-regular graph. A topological event happens when the planarity of $\mathcal{W}(G,t)$ is violated, including the case that two wavefront vertices meet. We interpret $\mathcal{W}(G,0)$ as the wavefront at time zero. While $\mathcal{W}(G,0)$ is geometrically overlapping with G, we assume that $\mathcal{W}(G,0)$ has — as a graph — the same topology as $\mathcal{W}(G,\epsilon)$ for a small $\epsilon > 0$. We observe that $\mathcal{S}(G)$ and G tessellate the plane into *faces* and that every face $f(e)$ belongs to an wavefront edge e: The face consists of all points in the plane which are swept by e, see Figure 3.

UNBOUNDED FACES AND INFINITE ARCS Note that certain faces will be unbounded since every point in the plane is eventually reached by the wavefront. By interpreting $\mathcal{S}(G)$ as a graph, it is reasonable to demand that each face corresponds to a cycle in $\mathcal{S}(G)$. Let us consider the last topological event that happened to the wavefront. After this event, the wavefront is a cycle that circumscribes G and all bounded faces of $\mathcal{S}(G)$. The vertices of this cycle trace the infinite arcs of $\mathcal{S}(G)$. Topologically, we add this cycle to $\mathcal{S}(G)$ and add the corresponding "infinite" arc to each "infinite" vertex of the cycle. Each "infinite" vertex has degree three. This interpretation of $\mathcal{S}(G)$ has practical advantages, e. g., when computing offset curves based on the straight skeleton, see Section 1.3.1.

ISOLATED VERTICES The definition of $\mathcal{S}(G)$ presented above assumes that G does not contain an isolated vertex. Aichholzer and Aurenhammer [AA96] mentioned that an isolated vertex could be approximated by a small straight-line segment. This must be done with some caution since $\mathcal{S}(G)$ does not continuously depend on G. For example, when multiple reflex arcs are incident to a point then a small perturbation of G would lead to a dramatically different straight skeleton $\mathcal{S}(G)$. See more on this issue in Section 2.5.3. However, one could simply define that an isolated vertex v emanates a wavefront which forms an axis-aligned square or any other polygon that has the property that its edges have an orthogonal distance of ϵ at time ϵ. That is, the supporting lines of the wavefront edges at time ϵ are tangential to the disk centered at v with radius ϵ. In fact, Demaine and O'Rourke [DO07] define the wavefronts emanated by an isolated vertex as an axis-aligned square in order to apply the straight skeleton for their fold-and-cut problems, see Section 1.3.3.

TERMINAL VERTICES In the same manner we could also alter the wavefront shape around terminal vertices. Consider a terminal vertex v and the two wavefront edges that emanate from the single incident edge e. Then we could basically place an arbitrary polygonal cap around v which connects the two wavefront edges from e. We only require that the segments of the cap have orthogonal distance ϵ at time ϵ. Nevertheless, using a rectangular cap — as introduced by Aichholzer and Aurenhammer — appears to be the most natural approach.

1.2.3 Roof and terrain model

Aichholzer et al. [AAAG95] presented an interpretation for the straight skeleton of simple polygons, which was extended to planar straight-line graphs by Aichholzer and Aurenhammer [AA96]. It turns out that this interpretation is a versatile tool in proofs of geometric properties of the straight skeleton. Further, it leads to one of the prominent applications of straight skeletons: roof construction and terrain modeling, see Section 1.3.2. The basic idea is to embed the wavefront propagation process in $\mathbb{R}^3$ in the following sense: the first two dimensions represent the plane spatial dimensions and the third dimension reflects the temporal dimension. The wavefront propagation now defines the so-called terrain $\mathcal{T}(G)$ of G as follows.

Definition 1.5 (terrain)**.** The *terrain* $\mathcal{T}(G)$ of G is defined by

$$\mathcal{T}(G) := \bigcup_{t \geq 0} \mathcal{W}(G, t) \times \{t\}. \tag{1.1}$$

Figure 4 illustrates the terrain $\mathcal{T}(G)$ of the graph G that is illustrated in Figure 3. Aichholzer et al. [AAAG95] used the term *roof* resp. *island* to indicate that $\mathcal{T}(P)$ of a simple polygon P has the following two interpretations. Firstly, $\mathcal{T}(P)$ can be interpreted as a particular roof of a house for which P models the footprint of the outer walls. Secondly, one can interpret P as the coastline of an island that has the shape of $\mathcal{T}(P)$. If the surrounding sea floods the island then the rising coastline has the shape of the rising wavefront in $\mathbb{R}^3$. In the case of planar straight-line graphs G Aichholzer and Aurenhammer [AA96] use the term *terrain* for $\mathcal{T}(G)$.

The terrain $\mathcal{T}(G)$ consists of plane facets that have a slope identical to the inverse of the propagation speed of the wavefront edges, which is 1. An edge of $\mathcal{T}(G)$ can either be convex or reflex. In the ordinary sense, we call an edge e of $\mathcal{T}(G)$ convex if the intersection of a small disk at any point in the relative interior of e with the points below $\mathcal{T}(G)$ is always convex. A reflex edge e of $\mathcal{T}(G)$ is defined likewise.

Definition 1.6 (reflex/convex arc, valley, ridge)**.** We call the arcs of $\mathcal{S}(G)$ which are traced out by reflex (convex) wavefront vertices *reflex arcs* (*convex arcs*). We call a reflex edge of $\mathcal{T}(G)$ a *valley* and a convex edge of $\mathcal{T}(G)$ a *ridge*.

Observation 1.7 ([AAAG95, AA98])**.** *The straight skeleton $\mathcal{S}(G)$ is the projection of the valleys and ridges of $\mathcal{T}(G)$ onto the plane $\mathbb{R}^2 \times \{0\}$. Moreover, the valleys correspond to the reflex arcs and the ridges correspond to the convex arcs.*

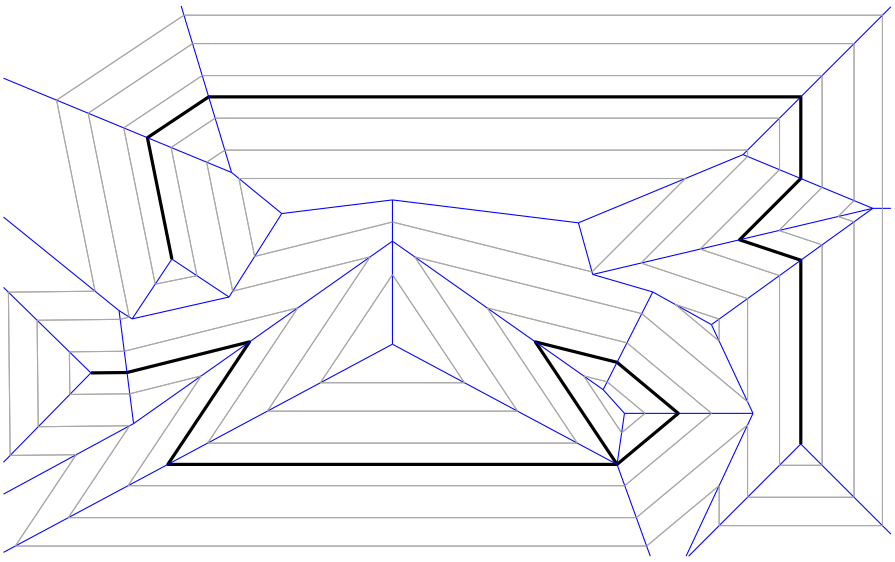


Figure 3: The straight skeleton $\mathcal{S}(G)$ (blue) of the planar straight-line graph G (bold). The wavefronts at three points in time are depicted in gray.

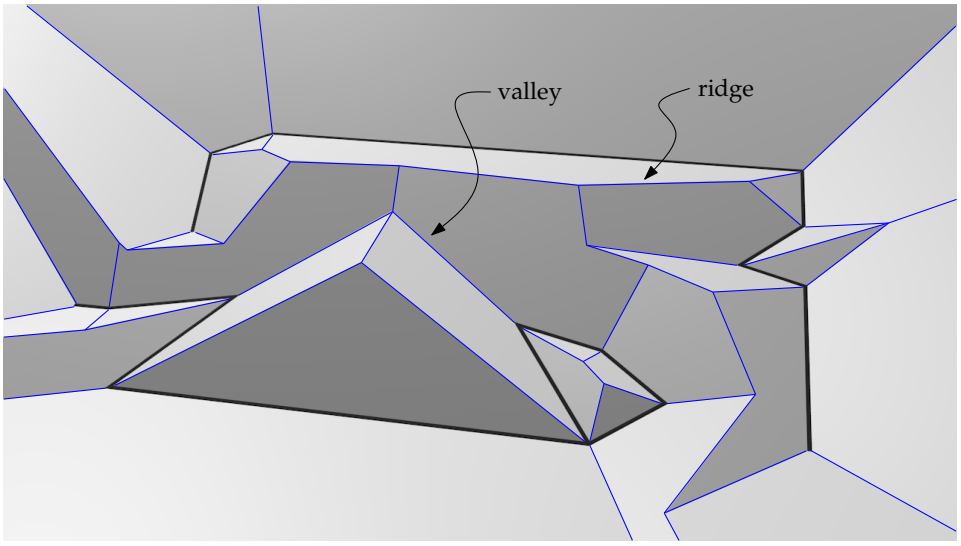


Figure 4: The terrain $\mathcal{T}(G)$ of the graph G which is illustrated in Figure 3. The ridges and valleys are in blue.

Following the notation of Cheng and Vigneron [CV07], we denote by $\hat{a}$ the edge of $\mathcal{T}(G)$ that corresponds to the arc a in $\mathcal{S}(G)$. Analogously, we denote by $\hat{f}(e)$ the facet of $\mathcal{T}(G)$ which corresponds to the face $f(e)$ of $\mathcal{S}(G)$.

ROOFS OF POLYGONS Aichholzer et al. [AAAG95] discussed the roof model of simple polygons P in more detail. They investigated more general roofs R on P which fulfill the property that each facet lies on a plane that contains an edge of P and has slope 1. The question arises whether such an R is equal to $\mathcal{T}(P)$. It turns out that this is not necessarily the case. However, R and $\mathcal{T}(P)$ are equal if all valleys of R are incident to P, or alternatively, if for any point $x \in R$ the path of the steepest descent leads to P. In other words, Aichholzer et al. [AAAG95] showed that among all roofs, $\mathcal{T}(P)$ has the peculiar property that it does not accumulate water when it is raining.

1.2.4 The motorcycle graph

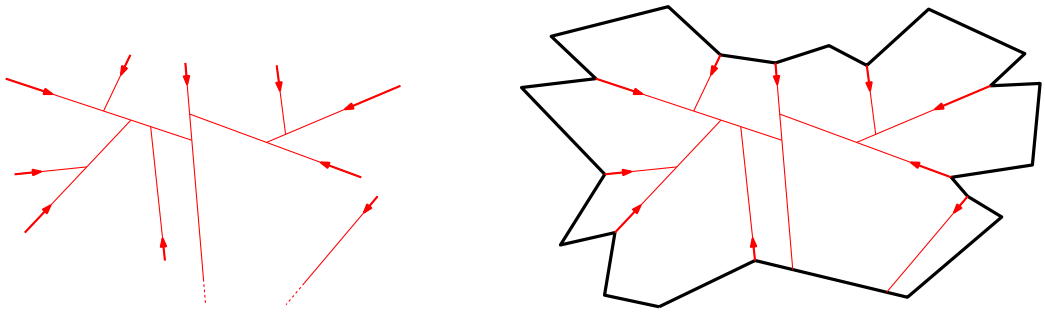
A straight-forward approach to computing the straight skeleton is through the simulation of the propagating wavefront. While edge events can be handled in a relatively efficient way the opposite holds for split events, see Section 1.4.2.1. It turns out to be non-trivial to efficiently determine which reflex wavefront vertex crashes into which wavefront edge. Let us consider for a moment only the simultaneous movement of the reflex wavefront vertices. In order to compute the straight skeleton it is important to know which reflex wavefront vertex is cutting off the trajectory of an other reflex vertex by reaching the common crossing point of their trajectories earlier. In order to extract this sub problem of computing straight skeletons, Eppstein and Erickson [EE99] introduced the so-called motorcycle graph.

A *motorcycle* is a point moving with constant speed on a straight line. Let us consider n motorcycles $m_1, \ldots, m_n$, where each motorcycle m_i has its own constant velocity $v_i \in \mathbb{R}^2$ and a start point $p_i \in \mathbb{R}^2$, with $1 \leq i \leq n$. The trajectory $\{p_i + t \cdot v_i \ : \ t \geq 0\}$ is called the *track* of m_i. While a motorcycle moves it leaves a *trace* behind. When a motorcycle reaches the trace of another motorcycle then it stops driving — it *crashes* —, but its trace remains. Note that it is possible that motorcycles never crash. Following the terminology of Eppstein and Erickson [EE99] such motorcycles are said to have *escaped*. It is easy to see that there can be up to $\binom{n}{2}$ intersections among the motorcycle tracks. However, no two motorcycle traces intersect in both interiors, which leads to at most n intersections among the traces.

Definition 1.8 (motorcycle graph)**.**) The *motorcycle graph* $\mathcal{M}(m_1, \ldots, m_n)$ of the motorcycles $m_1, \ldots, m_n$ is defined by the arrangement of the motorcycles traces.

In Figure 5 (a) we illustrate the motorcycle graph $\mathcal{M}(m_1, \ldots, m_{10})$ of ten motorcycles. Cheng and Vigneron [CV07] presented a straight-skeleton algorithm for simple polygons P, which is based on the motorcycle graph, see Section 1.4.2.4. They first define a motorcycle graph induced by a polygon. The idea is that every reflex vertex v of P defines a motorcycle which starts at v and has the same velocity as the wavefront vertex which corresponds to v.

NON-DEGENERACY ASSUMPTION Cheng and Vigneron [CV07] explicitly exclude the case that vertex events appear for the wavefront of P. The authors call this the *non-degeneracy*



(a) (b)

Figure 5: Left: The motorcycle graph $\mathcal{M}(m_1, \ldots, m_{10})$ of ten motorcycles. The velocities are represented by red arrows. Right: The motorcycle graph $\mathcal{M}(P)$, shown in red, induced by a simple polygon P (bold). Each reflex vertex of P emanates a motorcycle.

assumption. Eppstein and Erickson [EE99] remarked that perturbation techniques cannot be applied to transform these "degeneracies" to general cases. A small perturbation would change the straight skeleton drastically, see Section 2.5.3. On the other hand, such situations are very likely to occur, in particular if P contains collinear edges. Figure 1 shows a typical example. (We will be present a generalization of the motorcycle graph to arbitrary planar straight-line graphs in Section 2.5.) For the matter of simplicity we refer by the *non-degeneracy assumption* to the slightly more general assumption that no two motorcycles crash simultaneously into each other.[14]

In order to guarantee the correctness of the algorithm of Cheng and Vigneron [CV07], it is necessary to assume that the motorcycles *run out of fuel* when they reach an edge of P. We cover this circumstance by introducing the alternative concept of *walls*. We assume that the plane contains straight-line segments which model rigid walls. If a motorcycle reaches a wall then it crashes and its trace remains. Following our terminology, we define a motorcycle graph induced by a polygon by specifying the motorcycles and the walls.

Definition 1.9 (motorcycle graph induced by a simple polygon)**.** Let P denote a simple non-degenerate polygon. Each reflex vertex v emanates a motorcycle with the start point v and the same velocity as the corresponding wavefront vertex of v. Further, we consider the edges of P as walls. We denote by $\mathcal{M}(P)$ the resulting motorcycle graph and call $\mathcal{M}(P)$ the *motorcycle graph induced by* P.

Definition 1.10 (arm of a motorcycle)**.** Let m denote a motorcycle of $\mathcal{M}(P)$ that emanates from the vertex v. We call the two wavefront edges that are incident to the reflex wavefront vertex emanated from v, the *arms of* m. The one arm that is left of the track of m is called *left arm of* m and the other arm is called *right arm of* m.

We illustrate the motorcycle graph $\mathcal{M}(P)$ of a sample polygon P in Figure 5 (b). Cheng and Vigneron [CV07] also presented an extension of their algorithm to polygons with holes.

14 To be precise, Cheng and Vigneron assumed that no two valleys meet in a common point. Note that his assumption is slightly more general than the assumption that no vertex event happens: it can happen that two reflex wavefront vertices cause a multi split event such that no reflex vertex is emanated.

The motorcycle graph has to be extended accordingly: One has to introduce additional motorcycles at the convex vertices of the holes — which emanate reflex wavefront vertices within P — and one has to add the edges of the holes as walls, too.

1.3 APPLICATIONS

In the following section, we present several applications that appeared since the introduction of straight skeletons. The application of shape reconstruction by Oliva et al. [OPC96] even appeared at roughly the same time as straight skeletons and, in fact, the authors referred to the straight skeleton by a different name, namely *angular bisector networks* (ABN). In general, straight skeletons inherit many applications from Voronoi diagrams. Two typical applications — computing offset curves and terrain modeling — are immediately connected to the original definition of straight skeletons by Aichholzer et al. [AAAG95] and Aichholzer and Aurenhammer [AA96].

1.3.1 Mitered offset curves and NC-machining

Computational geometry has numerous applications in NC-machining. Computing offset curves is certainly one of the most important operations. Besides NC-machining, offset curves have a lot of further applications, like insetting/outsetting[15] paths in vector graphics editors and CAD software, computing tolerance domains around polygons or polygonal chains (e. g. for approximations within given bounds), computing curves parallel to a given curve (e. g. territorial domains on the sea defined by some fixed distance from a coastline), and so on.

First, we introduce some technical terms related to NC-machining. A workpiece that should be milled by an NC-machine is represented by a simple polygon P, possibly with holes. The tool in operation has the shape of a disk D_r with radius r and the origin as center. In the domain of NC-milling such a polygon P is often called *pocket* and the tool is called *cutter*. If we would move the center of the cutter along the boundary of P, we obviously remove too much material, see Figure 6 (a). The idea is to "shrink" the polygon P and then move the tool along the new boundary such that we obtain the desired workpiece, see Figure 6 (b). In mathematical terms, we want to determine a shape P' such that the Minkowski sum $P' + D_r := \{x + y \ : \ x \in P', y \in D_r\}$ equals P (except for certain portions, e. g., at the convex corners of P). The boundary of P' is called an *offset curve* of P with *offset distance* r. Note that there is only one continuous offset curve in Figure 6 (b) for that particular offset radius. If the tool did not leave the gap empty at the bottom of P then the bottom vertex of the hole would be cut off in this example.

At least two main questions related to computational geometry arise from the task of pocket machining: (i) how does one compute offset curves and (ii) how does one compute tool paths in order to remove the material in the interior of P? Held [Hel91] elaborated the computational problems related to pocket machining. He employs the Voronoi diagram of

15 Inkscape [Ink], a GPL-licensed vector graphics application, uses this terminology.

P in order to compute offset curves, where the offset curves are defined by the Minkowski difference $P - D_r$. Once the Voronoi diagram is available, this technique leads to a remarkably simple, efficient and robust method to compute offset curves, see [Hel91, HLA94]. Held [Hel91] also presented algorithms for two basic strategies in order to compute proper tool paths: *contour parallel* tool paths and *direction parallel* tool paths. Computing sophisticated tool paths is still a vital research area in these days. For example, Held and Spielberger [HS09] recently presented an algorithm that produces spiral tool paths which are suitable for high-speed machining. Computing an offset curve belongs to the first steps in their algorithm.

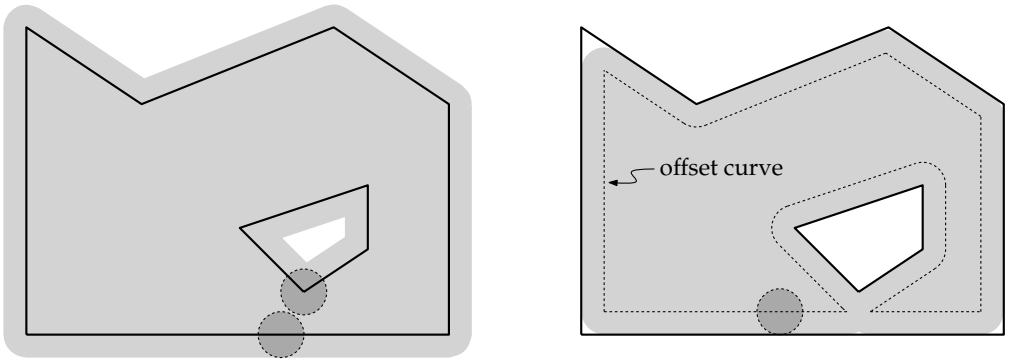
Voronoi-based offset curves consist of straight-line segments and circular arcs at the reflex vertices of P, see Figure 7 (a). Park and Chung [PC03] pointed out that such offset curves are undesirable if a high machining precision is required. The problem is that while the tool moves around a reflex vertex on a circular arc, it is permanently in touch with the reflex vertex, see Figure 7 (a). Oscillations of the workpiece or the tool during this time period lead to erosions of reflex vertices. Hence, they propose the so-called *mitered offset* curves, which are based on the straight skeleton instead of the Voronoi diagram. The basic idea is that sharp vertices of the polygon P should remain sharp in the offset curves, see Figure 7 (b). (Park and Chung [PC03] also mention that the offset curve based on the straight skeleton leads to long tool paths at sharp reflex vertices of the input. Hence, at reflex vertices with an interior angle of at least 1.5π the offset curves are trimmed by an additional line segment in order to shorten the tool path.)

The Voronoi diagram of simple polygons with holes — or more generally, of a planar straight-line graph — can be computed reliably in practice and implementations with an expected $O(n \log n)$ runtime exist, e. g., the Voronoi package VRONI by Held [Hel01]. In fact, VRONI also solves standard tasks for pocket machining, like computing offset curves, finding the maximum inscribed circle, or determining the medial axis[16]. Recently, Held and Huber [HH09a] extended the algorithms and the implementation behind VRONI to circular arcs. A nice byproduct of this extension is that the offset curves of straight-line polygons that are produced by VRONI, can serve as input to VRONI again.[17] As mentioned in the introduction of this chapter, the situation for the straight skeleton is clearly different. The only implementation available is the code by Cacciola [Cac04] within the CGAL library [CGA]. It accepts simple polygons with holes as input and exhibits a close-to quadratic runtime performance and memory footprint.

Park and Chung [PC03] circumvented the lack of efficient straight-skeleton algorithms by computing the mitered offset curves directly. Note that the mitered offset curve with offset distance t is equal to $\mathcal{W}(P, t)$. If no split event occurred until time t then one could simply compute the Voronoi diagram of P. After that one replaces the circular arcs of the offset curve by straight-line caps in order to obtain the mitered offset, without running into self-intersections of the offset curves. However, self intersections occur if the offset distance t is large enough such that a split event occurred until time t. Park and Chung [PC03] presented a relatively complex algorithm based on the concept of the so-called *pairwise interference detection*. Their algorithm basically puts offset segments parallel to the input edges and removes invalid portions of the offset curves afterwards.

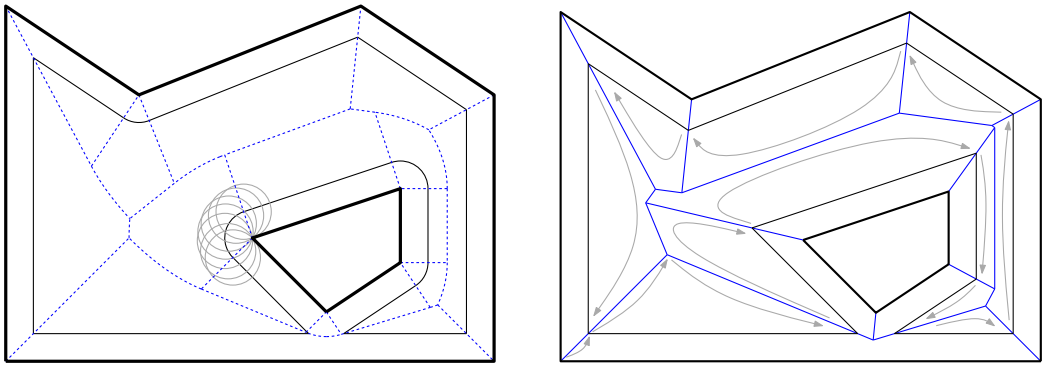
16 The medial axis is a certain subset of the Voronoi diagram.

17 The former version of VRONI was able to handle circular arcs by approximating using straight-line segments.



(a) (b)

Figure 6: Milling a simple polygon P with a hole (bold). Left: Simply tracing the boundary of P with a tool removes too much material (shaded). Right: Tracing the offset curve (dashed) removes the correct amount of material, except for some remaining material at convex corners.



(a) (b)

Figure 7: Two different approaches to obtain offset curves (thin) of the input (bold). Left: The approach based on the Voronoi diagram (dashed). The tool is shown in gray for several positions around are reflex vertex. Note that the tool remains in contact with the reflex vertex. Right: The approach based on the straight skeleton (dashed). The order, in which we traverse the straight skeleton to compute the offset curve, is sketched in gray.

Obviously, if the straight skeleton is available, the mitered offset curves are computed in an almost trivial way by simply traversing the straight skeleton in the very same fashion as it is done for Voronoi-based offset curves. In Figure 7 (b) we sketch the order in which the straight skeleton is traversed for a particular offset distance. After we start from an arbitrary input vertex, we basically follow the boundary of one incident straight-skeleton face after the other and stop at each time when we reach the desired offset distance on a straight-skeleton arc.

1.3.2 Building roofs and generating terrains

Automatic roof generation is an important task in 3D modeling, e. g., as part of an automatic city generator in a 3D modeling software. Let us consider a simple polygon P that represents the footprint of a building. How can we generate a realistic roof on top of the outer walls? A simple method to obtain a proper roof is to compute the roof model $\mathcal{T}(P)$ based on straight skeletons, see Section 1.2.3.

Laycock and Day [LD03] picked up this method and developed heuristic approaches in order to generate a larger variety of differently looking roofs. The original approach by interpreting the terrain $\mathcal{T}(P)$ above a simple polygon P as a roof results in the so-called *hip roof*. See Figure 8 for an example. Laycock and Day also describe how they obtain so-called *gable roofs*, *mansard roofs*, *gambrel roofs*, and *Dutch roofs*. For example, in order to generate a mansard roof, they consider the offset t until the first edge or split event happens and determine the offset at $0.85 \cdot t$, which is $\mathcal{W}(P, 0.85 \cdot t)$. The mansard roof consists of the original facets of $\mathcal{T}(P)$ which are restricted up to an height of $0.85 \cdot t$ and a top facet parallel to the ground plane. Straight skeletons became a general tool in order to generate sophisticated and realistically looking roofs, see [MWH+06, KW11, Hav05] for further examples.

The automatic generation of mountain terrains in the neighborhood of waters is very similar to roof construction. Assume we are given the shape of a river or a lake and we want to model the surrounding terrain. If the boundary of this shape is given by a planar straight-line graph G then the terrain $\mathcal{T}(G)$ gives a realistic model for the surrounding terrain. In Figure 9 we show the generated terrain that illustrates the an part of the river Danube called "Schlögener Schlinge" in Austria. The generated terrain gives indeed a good approximation of the actual real-world scene.

The 3D models of Figure 8 and Figure 9 were generated by our straight-skeleton implementation Bone, see Section 2.5.3. Our implementation can be used to export the terrain $\mathcal{T}(G)$ of a planar straight-line graph G in a file, which is further processed by the free 3D modeling software Blender [Ble].

1.3.3 Mathematical origami and the fold-and-cut problem

Straight skeletons possess an interesting application in the field of mathematical origami: the *fold-and-cut problem*. Let us consider a simple polygon P drawn on a sheet of paper. (The polygon P could illustrate a flower or an animal.) We are allowed to apply a sequence of folds of the paper along straight lines. Finally, we take a scissor and cut the folded paper along a straight line into pieces. Which sequence of folds and which line for the final cut

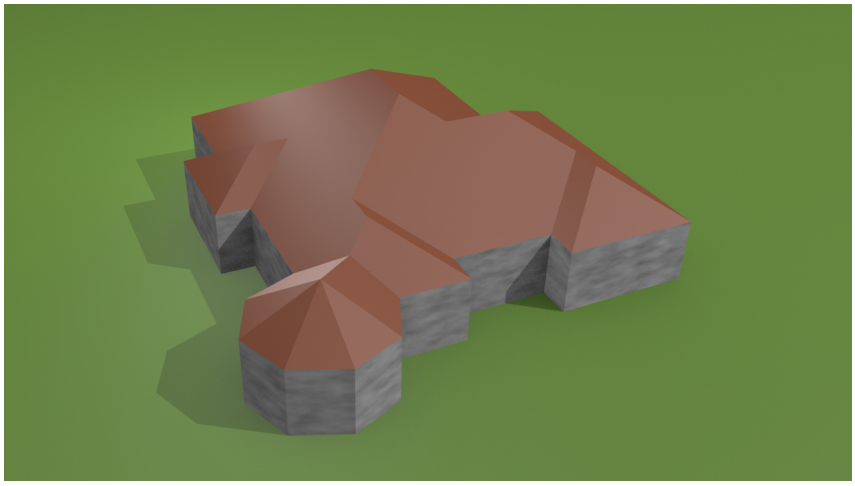


Figure 8: A hip roof generated by our straight-skeleton implementation Bone from the polygon that forms the footprint of the walls. Every facet of this roof has an identical slope.

Figure 9: A terrain generated from the boundary of the blue river by our straight-skeleton implementation Bone. The figure illustrates the "Schlögener Schlinge"', which is a part of the river Danube in Austria (48°26′ 10″ N 13°51′ 50″ E). The boundary of the river is based on data obtained from OpenStreetMap [OSM].

do we have to apply in order to obtain a piece of paper that has the shape P? This problem, among others, is discussed in the book by Demaine and O'Rourke [DO07] in an illustrative and detailed fashion. They summarize the fold-and-cut problem as follows: (i) which shapes can be produced by such a fold-and-cut sequence and (ii) how can we compute a corresponding fold-and-cut sequence if the shape is given?

As described in [DO07], Chapter 17, the basic idea is to align the edges of P along straight lines by folding according to a so-called *crease pattern*. More precisely, the question whether a given polygon can be produced by a fold-and-cut sequence is equivalent to the question whether a crease pattern exists such that the edges of P can be arranged on a straight line and that any other point of the paper does not lie on this straight line. A cut through this line cuts exactly at the edges of P. The *universality result* states that every planar straight-line graph G can be cut by an appropriate fold-and-cut sequence [DDL98].

Demaine et al. [DDL98] presented an algorithm to compute crease patterns based on the straight skeleton. The input to their algorithm is a planar straight-line graph G. The motivation to use straight skeletons is that folding a paper at the bisector of two edges aligns both edges on a common straight line. Note that the arcs of the straight skeleton lie, by definition, on the bisectors of the defining pair of edges of G. The basic idea is that the straight skeleton almost poses an appropriate crease pattern (depending whether an arc is reflex or convex, the paper is folded in one or the other direction). As elaborated in [DO07], additional creases, so-called *perpendicular creases*, need to be introduced to obtain the final crease pattern for a given input graph G. Demaine et al. [DDL98] proved that a certain class of planar straight-line graphs can be obtained with a single cut using their crease-pattern algorithm based on straight skeletons. Crease patterns for arbitrary planar straight-line graphs can be computed using an algorithm based on disk packings. However, the crease patterns based on straight skeletons tend to be simpler [DO07].

Straight skeletons have also been applied to further problems in the field of mathematical origami and also to polyhedral wrapping problems, see [DDM00] for example.

1.3.4 Shape reconstruction and contour interpolation

Oliva et al. [OPC96] introduced an alternative version of the medial axis, which they called *angular bisector network* (ABN) and which turns out to be exactly the straight skeleton. Their motivation for this skeleton structure originates from the problem of 3D-surface reconstruction, for which they need to have a skeleton comprising straight-line segments only. They consider a sequence of cross-sections of a 3D-surface that lie on parallel planes. Each cross section consists of nested non-intersecting polygons defining so-called material and non-material domains. The problem is to compute a reconstruction of the original 3D-surface based on the set of cross sections. This is a typical problem in medical imaging.

The problem mentioned above is considered to be difficult in the presence of complex branches of the surface. Oliva et al. [OPC96] interpret the problem as an interpolation task between two consecutive sections. In the first step the polygonal shapes on two consecutive sections are projected on a parallel plane. Then they consider the symmetric difference of both shapes. This difference consists of nested polygons and each polygon comprises edges of both cross sections. In the next step they compute the straight skeleton of the difference

shape and build a triangulation of the straight-skeleton faces. This triangulation is lifted to 3D in order to pose a patch of the 3D-surface between the two cross sections. The nodes of the straight skeleton lie on an intermediate layer between the two cross sections.

Barequet et al. [BGLSS04] presented a very similar approach. In contrast to [OPC96], they consider a different triangulation scheme and assign different heights to the vertices between the cross sections.

1.3.5 Polygon decomposition

Tănase and Veltkamp [TV04b] proposed an approach to polygon decompositions based on straight skeletons, where the split events define how the polygon is decomposed into sub polygons. They consider a reflex wavefront vertex v that led to a split event by crashing into the wavefront edge e. Then the polygon is decomposed by (i) the arc a that is traced out by v and (ii) the projection line of the endpoint of a that is orthogonal to e, until the boundary of $f(e)$ is hit. These two paths can be interpreted in the terrain model as two possible paths of a raindrop that starts at the endpoint of the lifted arc $\hat{a}$, see Section 1.2.3. Decompositions of this type form the first phase of their algorithm. In the second phase, further local simplifications of the decomposition are applied.

The decomposition steps of the first phase are similar to the randomized partitioning used by Cheng and Vigneron [CV07]. However, Cheng and Vigneron [CV07] do not investigate polygon decomposition schemes per se, but used their partition in order to devise a randomized straight-skeleton algorithm, see Section 1.4.2.4.

1.3.6 Area collapsing in geographic maps and centerlines of roads

In geographic information systems a common task is the simplification of maps by collapsing certain areas. Assume we are given a map of a landscape, including rivers, streets and municipalities. The rivers and streets are given as polygonal areas. The problem of area collapsing asks for a method to collapse a certain polygonal area A (e. g., a river or a street) to a one-dimensional structure by dividing A among its neighboring areas.

Haunert and Sester [HS08] introduced a method based on the straight skeleton $\mathcal{S}(A)$ of the polygon A. Each straight-skeleton face $f(e)$ is merged with the neighboring area that shares the edge e with A. In order to assign larger portions of A to larger neighboring areas, they employ the weighted straight skeleton, cf. Section 1.5.2. The weighted straight skeleton allows them to adjust the sizes of the straight-skeleton faces in A accordingly. Their method respects certain topological constraints, e. g., it maintains connectivity. For instance, if a river joins a lake then it is desirable that the collapsed river is still connected to the lake.

Haunert and Sester [HS08] also discuss the problem of computing the centerlines of roads. They consider a road network of a city, where each road is given by the polygonal area it occupies. How can one obtain a one-dimensional representation such that each road is represented by a polygonal chain? Haunert and Sester [HS08] compute the so-called centerline of a road by, roughly speaking, considering those straight-skeleton arcs which are not defined by two adjacent boundary edges of the road area. Special care is taken in the

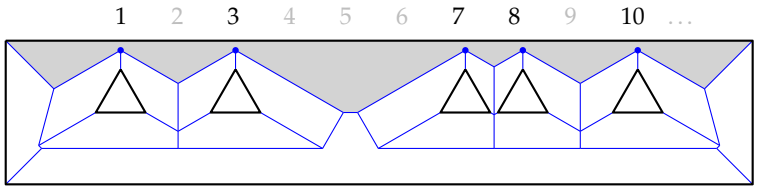


Figure 10: Sorting natural numbers can be reduced to the computation of the straight skeletons (blue) of a polygon with holes (bold). The nodes that are depicted by small disks correspond to the input numbers $\{1, 3, 7, 8, 10\}$.

presence of road junctions. If three or more roads meet in a junction it is rather unlikely that the four centerlines meet in a common straight-skeleton node. Haunert and Sester [HS08] presented a heuristic approach to detect such junctions and to connect the centerlines of the joining roads with a common junction node.

1.4 PRIOR WORK

1.4.1 Runtime bounds for the straight skeleton

To the best of our knowledge, the currently best known lower bound to compute the straight skeleton of a simple polygon with n vertices is $\Omega(n)$. In the convex case a simulation of the propagating wavefront gives us an almost optimal algorithm using $O(n \log n)$ time, see Section 1.4.2.1. For the more general case of monotone polygons, Das et al. [DMN+10] presented an $O(n \log n)$ algorithm. However, for arbitrary simple polygons, no algorithm that is even reasonably close to linear is known. Currently, the fastest algorithm is due to Eppstein and Erickson [EE99] with a worst-case runtime of $O(n^{17/11+\epsilon})$. This poses a high contrast to the situation for Voronoi diagrams, for which Chin and Snoeyink [CSW99] presented an optimal linear-time algorithm. Note that in the convex case the Voronoi diagram and the straight skeleton are coinciding and the algorithm by Chin and Snoeyink solves the straight-skeleton problem in optimal time as well. However, for convex polygons also a simpler Voronoi algorithm is known due to Aggarwal et al. [AGSS87].

For planar straight-line graphs and for polygons with holes, the best known lower bound is $\Omega(n \log n)$. This is easy to see, since we can simply reduce the sorting problem to straight skeletons. Assume n distinct natural numbers $a_1, \ldots, a_n$ are given, which have to be sorted. For any a_k, we place a small equilateral triangle that is hinged with its top vertex at the coordinates $(a_k, 0)$. The length of the edges of the triangles are considered to be less than 1, say 0.9. Then we put a box around the triangles such that the top vertices of the box have small y-coordinates, say 0.1. The box and the triangles form a simple polygon P with holes with $3n + 4$ vertices and can be constructed in $O(n)$ time. After computing $\mathcal{S}(P)$ we consider the face $f(e)$ of the top edge e of P that is within P. It is easy to see that the nodes that appear as reflex vertices of $f(e)$ correspond to the input $a_1, \ldots, a_n$ and occur in sorted order along the x-axis, see Figure 10. A simple traverse of $f(e)$ gives us the sorted sequence in linear time.

1.4.2 Algorithms for computing straight skeletons and motorcycle graphs

1.4.2.1 *Aichholzer et al., 1995*

When Aichholzer et al. [AAAG95] introduced straight skeletons of simple polygons P, they also presented an algorithm, which simulates the propagation of the wavefront in a discrete manner. Their algorithm maintains a priority queue Q which contains all potential edge events for each wavefront edge of P, prioritized by their occurrence time. Assume for a moment that P is convex, which means that only edge events occur. The algorithm fetches the earliest event from Q and processes it accordingly. That is, the incident vertices of the affected edge e are merged to a single vertex v. Note that the velocity of v is given by the two incident wavefront edges. Hence, the collapsing times of the two incident edges of v got invalid and one has to (i) re-compute them and (ii) update the priority queue accordingly. A single edge event affects only a constant number of entries within Q and can therefore be handled in $O(\log n)$ time. For convex polygons P this algorithm computes the straight skeleton in $O(n \log n)$ time.

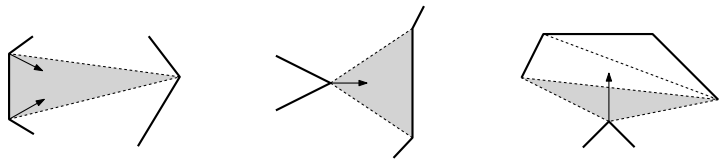
In the presence of split events the situation gets more complicated. Efficiently determining the first split turns out to be non-trivial. Let v denote a reflex vertex of P. The problem is to determine in which wavefront edge the corresponding reflex wavefront vertex will crash. Simple ray-shooting does not solve the problem. The wavefront edge e, which is hit by the ray, could move off the ray during the wavefront propagation. Maintaining the wavefront edge that is hit by the ray does not solve the problem either: another reflex wavefront vertex from aside could cross the ray at any position. However, detecting such incidents seems to be costly.

The idea of Aichholzer et al. [AAAG95] is the following: Consider two consecutive edge events at time t' and $t > t'$ and assume that the wavefront is free of self-intersections until time t'. The wavefront becomes self-intersecting until t if and only if one or more split events happened in the time interval $[t', t]$. More importantly, Aichholzer et al. [AAAG95] were able to show that these split events can be determined from the wavefront $\mathcal{W}(P, t)$. In particular, they showed that the corresponding split events can be found and processed in $O(n \log n)$ time.

Testing whether a self-intersection is present at the k-th edge event is done in linear time using the triangulation algorithm by Chazelle [Cha91]. Applying exponential searching, the first split event can be determined at the costs of $O(\log k)$ intersection tests, i. e., in $O(n \log k)$ time, where the first split event happened just before the k-th edge event. Applying this algorithm recursively on each sub-polygon of $\mathcal{W}(P, t)$ after the first split event leads to a total runtime of $O(n^2 \log n)$. Denoting by $r \in O(n)$ the number of reflex vertices, one can further refine the runtime analysis to $O(nr \log n)$.

1.4.2.2 *Aichholzer and Aurenhammer, 1996*

Aichholzer and Aurenhammer [AA96, AA98] presented an algorithm for planar straight-line graphs which is based on a kinetic triangulation. Their algorithm accepts a planar straight-line graph G with n vertices as input. The basic idea is again to simulate the prop-



(a) edge event (b) split event (c) flip event

Figure 11: The three types of topological changes during the propagation of the triangulation.

agating wavefront, but to exploit the topological changes in a kinetic triangulation in order to determine the edge and split events.

The algorithm starts with an initial triangulation of G and keeps the area $\bigcup_{t' \geq t} \mathcal{W}(G, t')$ triangulated for all times $t \geq 0$. Aichholzer and Aurenhammer [AA96] also include infinite triangles to the triangulation such that the entire plane is triangulated at time zero. During the propagation of the wavefront, the triangulation keeps the area $\bigcup_{t' \geq t} \mathcal{W}(G, t')$ triangulated and at certain points in time the triangulation may change its topology: triangles collapse to a point or to a line. Whenever such an event occurs local modifications have to be applied in order to maintain a proper triangulation. The essential observation is that edge and split events of the wavefront correspond to a collapse of a triangle. Unfortunately, not every triangle collapse corresponds to an event of the wavefront. Aichholzer and Aurenhammer [AA96] distinguish the following types of events, as illustrated in Figure 11.

- **Edge event**: A triangle collapsed due to an edge event of the wavefront. The corresponding edge e collapsed to zero length and the one triangle having e as an edge collapsed with it.
- **Split event**: A triangle collapsed because a split event in the wavefront happened. In the simple case a reflex wavefront vertex moves into a wavefront edge e and the one triangle having e as an edge collapses thereby. In case of a multi split event two or more reflex wavefront vertices meet in a point and again certain triangles collapse.
- **Flip event**: A triangle collapsed because a wavefront vertex crosses an inner triangulation diagonal. The affected diagonal has to be flipped[18] in the triangulation in order to maintain a valid triangulation. This event does not immediately correspond to a topological change of the wavefront.

The algorithm of Aichholzer and Aurenhammer computes $\mathcal{S}(G)$ by keeping track of the topological changes within the triangulation. This again involves a priority queue Q containing the events. The runtime complexity depends on the number of events that occurred. The number of edge and split events corresponds to the number of nodes of $\mathcal{S}(G)$. According to Lemma 2.4, this number[19] is in $O(n)$. Each edge and split event involves the adaption of a certain number of wavefront vertices and, as a consequence, the re-calculation of the collapsing times of up to $O(n)$ triangles that are incident to these vertices. Summarizing, all edge and split events can be handled in $O(n^2 \log n)$ time.

18 If we remove the considered edge just before the flip event happens then we get a convex quadrilateral which can be triangulated in two ways. By an edge flip we mean that we triangulate the quadrilateral the other way.

19 Actually, in order to prove this Lemma, Aichholzer and Aurenhammer count the number of triangles in the triangulations, see the proof of Lemma 2.4.

However, finding a sophisticated upper bound for the number of flip events appears to be harder. First of all, a single flip event is handled in $O(\log n)$ time since it only involves the two triangles that share the affected triangulation diagonal. The upper bound of $O(n^3)$ for the number of flip events is relatively easy to see. Each wavefront vertex moves along a straight-line with constant speed. Consider the vertices p, q, r of a triangle Δ and denote by $p(t), q(t), r(t)$ their positions at time t, respectively. We denote by $u, v, w \in \mathbb{R}^2$ the velocities of p, q, r. The triangle Δ collapses if and only if the points p, q, r get collinear. That is

$$\begin{vmatrix} p_x(t) & q_x(t) & r_x(t) \\ p_y(t) & q_y(t) & r_y(t) \\ 1 & 1 & 1 \end{vmatrix} = \begin{vmatrix} p_x(0) + t \cdot u_x & q_x(0) + t \cdot v_x & r_x(0) + t \cdot w_x \\ p_y(0) + t \cdot u_y & q_y(0) + t \cdot v_y & r_y(0) + t \cdot w_y \\ 1 & 1 & 1 \end{vmatrix} = 0, \tag{1.2}$$

where the x- and y-coordinates are denoted by subscriptions. This quadratic equation in t is fulfilled for exactly zero, one, two or all values of t. As a consequence, a single triangle with vertices p, q, r collapses at most twice. Since we have $\binom{n}{3}$ possible triples of vertices the number of flip events is bounded by $O(n^3)$. Summarizing, the runtime complexity of the algorithm is in $O((n^2 + k) \log n)$, where $k \in O(n^3)$ denotes the number of flip events.

Interestingly, to the best of our knowledge, no input is known that exceeds a quadratic number of flip events. We will revisit the open question concerning this gap of a linear factor in Section 2.2. For some typical input data Aichholzer and Aurenhammer [AA96] observe an actual runtime close to $O(n \log n)$. However, no published runtime statistics are known.

1.4.2.3 *Eppstein and Erickson, 1999*

Eppstein and Erickson [EE99] presented a straight-skeleton algorithm for simple polygons, which is applicable to planar straight-line graphs and which can even be extended to compute the weighted straight skeleton, cf. Section 1.5.2. Let us recall the discussion in Section 1.4.2.1 concerning the determination of the split events. As elaborated by Aichholzer et al. [AAAG95], a simple ray-shooting from reflex vertices does not solve the problem of finding the next split event. However, one could consider for every propagating wavefront edge e the triangle (possibly infinite in size) that is bound by e at time zero and the trajectories of its incident wavefront vertices. The idea is to keep track of those pairs of reflex vertices v and wavefront edges e, where the potential split event of v and e takes place within the corresponding triangle of e. The next split event is given by closest pair in terms of the earliest split-event time.

Eppstein and Erickson [EE99] pursued this approach by employing very powerful closest-pair data structures. First of all, Eppstein and Erickson lift the problem to $\mathbb{R}^3$ in the same fashion as it is done for the terrain model $\mathcal{T}(G)$. Every reflex vertex v of $\mathcal{W}(G, 0)$ defines a ray in $\mathbb{R}^3$ starting at v and supports the corresponding ridge in $\mathcal{T}(G)$. At every initial wavefront edge e a corresponding (lifted) triangle, which supports $\hat{f}(e)$, is considered. The two other edges of the triangle are given by the supporting rays of the lifted trajectories of the incident wavefront vertices, see Figure 12. Note that if we consider the unweighted straight skeleton then the triangles have slope 1, whereas the rays have a slope of at most 1.

Eppstein and Erickson [EE99] consider a sweep plane, which sweeps $\mathbb{R}^3$ along the z-axis. Whenever the sweep plane reaches the top of a triangle, an edge event happens. When the intersection of a ray with a triangle is reached a split happened, including multi split

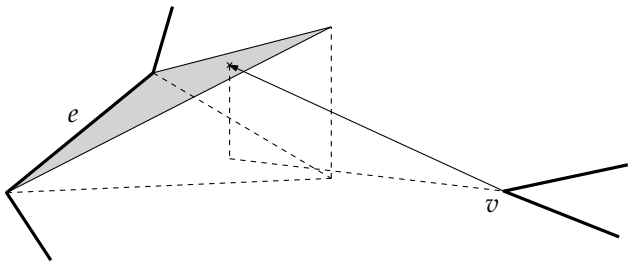


Figure 12: The triangle of an edge e and a ray of the reflex vertices v, as considered by the algorithm by Eppstein and Erickson [EE99]. The lowest intersection of a ray with a triangle gives the next split event.

events. For every edge and split event a constant number of triangles and rays is removed and added. Every multi split event involving k reflex wavefront vertices leads to the deletion and insertion of $O(k)$ triangles and rays. Eppstein and Erickson [EE99] maintain the triangles in a priority queue Q, prioritized by the height of their top vertex. The total costs for maintaining Q is in $O(n \log n)$. The essential part of the algorithm consists of determining the ray-triangle intersections.

Central aspects in the algorithm by Eppstein and Erickson [EE99] are range searching techniques that allow the application of time-space trade-offs. Assume we are in possession of a data structure that supports fast queries but has a high space consumption and, conversely, a data structure that has high query times but low space bounds. The idea is to mix those two data structure in a hierarchical fashion in order to obtain characteristics for query time and space consumption between the two original data structures. A survey by Agarwal and Erickson [AE99] discusses these techniques in detail.

Eppstein and Erickson [EE99] denote by s the space required by a data structure and express the time complexity as a function in s. One step towards their straight-skeleton algorithm is a data structure which itself is based on several other data structures and allows lowest intersection queries among n rays and triangles within $O(n^{1+\epsilon}/s^{1/4})$ time, for any $\epsilon > 0$. Another main ingredient is a data structure by Agarwal and Matoušek [AM94] in order to answer ray shooting queries, again in $O(n^{1+\epsilon}/s^{1/4})$ time. By balancing time and space complexities, Eppstein and Erickson finally obtain a straight-skeleton algorithm which uses $O(n^{8/5+\epsilon})$ time and space. They further note that this algorithm is also applicable in order to compute the weighted straight skeleton.

For the unweighted case, Eppstein and Erickson [EE99] prove Theorem 2.7 in order to devise a slightly faster algorithm, see Section 2.1. In this case, all triangles of the algorithm have identical slope 1. This fact and the theorem mentioned above can be exploited in order to use more efficient data structures for intersection queries between rays and triangles. Using these refinements the algorithm by Eppstein and Erickson [EE99] has a theoretical worst-case time and space complexity of $O(n^{1+\epsilon} + n^{8/11+\epsilon} r^{9/11+\epsilon})$ for the unweighted straight skeleton of planar straight-line graphs, where r denotes the number of reflex vertices.

Eppstein and Erickson also present an approach that uses a simpler data structure by Eppstein [Epp00] in order to maintain the closest-pair between a ray and a triangle. This data structure uses a quadtree structure on the matrix of the pairwise distances. Using this

data structure, the weighted straight skeleton can be computed in $O(n \log n + nr)$ time and $O(nr)$ space and the unweighted straight skeleton can be computed in $O(n \log n + nr)$ time and $O(n + r^2)$ space. However, no implementations of these algorithms are known and, in particular, no runtime statistics have been published.

Similar techniques as sketched above are also used by Eppstein and Erickson [EE99] in order to compute the motorcycle graph of n motorcycles $m_1, \ldots, m_n$. The motorcycle graph was introduced in their paper in order to describe the essential sub problem of computing straight skeletons, see Section 1.2.4. The idea is that each motorcycle m_i defines a tilted ray in $\mathbb{R}^3$ and a vertical curtain above the ray. The intersection of a ray of the motorcycle m_i with a curtain of a motorcycle m_j corresponds to a crash of m_i into the trace of m_j. Their algorithm runs in $O(n^{17/11+\epsilon})$ time and space.

1.4.2.4 *Cheng and Vigneron, 2002*

Cheng and Vigneron [CV02, CV07] were the first to exploit the relationship between motorcycle graphs and straight skeletons in order to compute the straight skeleton. Their straight-skeleton algorithm accepts non-degenerate polygons P with holes as input.

First of all, Cheng and Vigneron compute the motorcycle graph $\mathcal{M}(P)$ induced by P. Let $m_1, \ldots, m_n$ denote n motorcycles. Further, we denote by p_i the start point and by v_i the velocity of the motorcycle m_i. They observed that the tracks, on which the n motorcycles drive, can have $\Theta(n^2)$ pairwise intersections, but only $O(n)$ among them actually correspond a crash. The challenge is to geometrically separate motorcycles that do not interact. At the first sight this appears to be difficult, since a single motorcycle can be very fast and cross the tracks of many motorcycles. The central idea of Cheng and Vigneron is to employ so-called ϵ-cuttings.

Chazelle [Cha04] explains ϵ-cuttings as follows. Let H denote a set of n hyperplanes in $\mathbb{R}^d$. An ϵ-cutting is a tessellation of $\mathbb{R}^d$ into non-overlapping simplices such that each simplex intersects at most ϵn hyperplanes of H. Chazelle [Cha93] presented an algorithm that runs in optimal time $O(n\epsilon^{1-d})$ and constructs a cutting of optimal size $O(\epsilon^{-d})$.

Cheng and Vigneron [CV07] use $1/\sqrt{n}$-cuttings in $\mathbb{R}^2$ on the supporting lines of the motorcycle tracks. Hence, every triangle of the cutting intersects at most $\sqrt{n}$ motorcycle tracks. The cutting of $O(n)$ size can be computed in $O(n\sqrt{n})$ time by Chazelle's algorithm [Cha93]. Once the cutting is computed, Cheng and Vigneron [CV07] simulate the movement of the motorcycles within the cutting. They distinguish two types of events:

- Switch event: A motorcycle reaches the boundary of a cutting-triangle and migrates to a neighboring triangle.
- Crash[20] event: A motorcycle crashes into the trace of another motorcycle.

The motorcycle graph can be computed independently within each cell. A priority queue Q contains each event and the events are processed in chronological order. First, they compute all potential switch events in advance and insert them all into the priority queue Q in $O(n\sqrt{n} \log n)$ time. Note that a single motorcycle may indeed cross $\omega(\sqrt{n})$ simplices, but the total number of switch events among all motorcycles is in $O(n\sqrt{n})$.[21] In order to man-

20 The original phrasing of Cheng and Vigneron was "impact event".

21 By $\omega(f(n))$ we denote the set $\Omega(f(n)) \setminus \Theta(f(n))$ following the notation of $o(f(n))$ versus $O(f(n))$.

age the potential crash events, Cheng and Vigneron maintain for each cutting-triangle an arrangement on the tracks of each motorcycle that is or has been visiting the triangle. Initially, each arrangement in a cutting triangle C is built for the motorcycles that start within C. Using a standard plane-sweep algorithm — e. g., Bentley-Ottmann [BO79] — this can be achieved in $O(n\sqrt{n}\log n)$ time.

When a switch event occurs for the motorcycle m_i they first check whether m_i has already crashed. If it is still driving they insert the track of m_i into the arrangement of the new triangle C. Cheng and Vigneron use a binary tree structure on each cell in each arrangement, which enables them to insert the track of m_i in $O(k \log n)$ time, where k denotes the number of new arrangement vertices introduced. For each such arrangement vertex they add a potential crash event into Q, depending on which of both motorcycles involved reaches the vertex earlier. Cheng and Vigneron argue that the total number of arrangement vertices is bounded by $O(n\sqrt{n})$. In the triangle C, they consider an arrangement vertex v on the intersection of the tracks of m_i and m_j. Either m_i or m_j started or crashed within C. Cheng and Vigneron charge m_i with v, if m_i started or crashed within C and likewise for m_j. In order to bound the total number of arrangement vertices they count for each motorcycle m_k the number of vertices for which m_k has been charged. In order to do so they only have to consider the cutting triangles where m_k started resp. crashes. We get at most $2\sqrt{n}$ vertices for each motorcycle and $O(n\sqrt{n})$ in total. Hence, all switch events can be handled in $O(n\sqrt{n}\log n)$ time.

When a crash event is processed for m_i, Cheng and Vigneron check whether m_i has not crashed already. If it is still driving, the motorcycle m_i definitely crashed at the arrangement vertex for which the current crash event has been created. The total number of potential crash events is bounded by the total number of arrangement vertices, which is $O(n\sqrt{n})$, as elaborated above. A single crash event is easily handled in $O(\log n)$ time and hence the total costs for all crash events is in $O(n\sqrt{n}\log n)$.

The space complexity for all arrangements is bounded by $O(n\sqrt{n})$. Summarizing, Cheng and Vigneron compute the motorcycle graph of n motorcycles in $O(n\sqrt{n}\log n)$ time and $O(n\sqrt{n})$ space[22]. This is a slight improvement compared to the algorithm by Eppstein and Erickson [EE99].

Strictly speaking, the discussed algorithm computes $\mathcal{M}(m_1, \ldots, m_n)$ but does not consider the edges of P as walls, i. e., they do not compute $\mathcal{M}(P)$ as defined in Section 1.2.4. However, Cheng and Vigneron note that their algorithm can handle motorcycles that *run out of fuel* when they reach the boundary of P. In order to bound the length of the motorcycle traces accordingly, one could apply the ray-shooting algorithm within a polygon by Chazelle et al. [CEG+91], which supports a single ray shooting in $O(\log n)$ time.

Cheng and Vigneron [CV07] also mention a randomized and simpler version of their algorithm, where the supporting lines of the traces of $\sqrt{n}$ randomly chosen motorcycles could be used instead of an $1/\sqrt{n}$-cutting. Furthermore, they mention a simpler cutting algorithm in the plane by Har-Peled [HP00]. Nevertheless, no implementations or runtime statistics are known for their motorcycle graph algorithm.

In order to compute the straight skeleton of a simple non-degenerate polygon P with n vertices, Cheng and Vigneron [CV07] exploit Theorem 2.11. The fact that $\mathcal{M}(P)$ covers

22 Cheng and Vigneron did not discuss space consumption in their work.

the reflex arcs of $\mathcal{S}(P)$ allows them to devise a randomized algorithm for the computation of $\mathcal{S}(P)$. In a nutshell, the algorithm chooses random sites on the straight skeleton which induce a partition of P. The idea is to compute $\mathcal{S}(P)$ on each part of the partition in a divide-and-conquer fashion, as described in the following.

They first choose a point p on a convex arc of $\mathcal{S}(P)$ and project p vertically onto the terrain $\mathcal{T}(P)$. The projection point $\hat{p}$ lies on a ridge of $\mathcal{T}(P)$ and defines two or three paths on $\mathcal{T}(P)$ that follow the steepest descent and correspond to the raindrop traces in Section 1.2.3. Cheng and Vigneron consider a set E of such ridge points and consider the vertical projections of the corresponding raindrop paths onto the plane. These projected paths induce a partition of P, which Cheng and Vigneron call *canonical partition of P induced by E*. (Note that these paths do not cross, but may merge in a valley of $\mathcal{T}(P)$.) Cheng and Vigneron observed that the canonical partition of P induced by E can be computed recursively. Consider a polygon that is partition by E_1 and a cell C of that partition that is further partitioned by E_2. They proved that the resulting partition is equal to the partition of P induced by $E_1 \cup E_2$.

Cheng and Vigneron presented a method to compute the intersection points $L \cap \mathcal{S}(P)$, for a line L parallel to the y-axis, by computing a vertical slice of $\mathcal{T}(P)$ above L. The resulting points E are used in order to subdivide P according to the partition scheme described above. Cheng and Vigneron keep on subdividing cells of the partition scheme using this method. Once the cells have reached a certain size, they compute the straight skeleton on this cell by a brute force method. The way how Cheng and Vigneron select the vertical line L involves randomness. Finally, Cheng and Vigneron are able to compute the straight skeleton of a simple non-degenerate polygon P in expected $O(n \log^2 n)$ time if the motorcycle graph is already given. Since the motorcycle graph $\mathcal{M}(P)$ can be computed in $O(r\sqrt{r} \log r)$ time they obtain a straight-skeleton algorithm that runs in $O(n \log^2 n + r\sqrt{r} \log r)$ time.

Cheng and Vigneron [CV07] extended their algorithm to non-degenerate polygons with holes. If h denotes the number of holes their algorithm computes the straight skeleton in $O(n\sqrt{h} \log^2 n + r\sqrt{r} \log r)$ expected time.

1.4.2.5 *Felkel and Obdržálek, 1999*

Felkel and Obdržálek [FO99] briefly presented a wavefront-type straight-skeleton algorithm for simple polygons that is based on the algorithm of Aichholzer et al. [AAAG95]. As discussed in Section 1.4.2.1, the simulation of the edge events is simple, but handling split events is the challenge for a wavefront-type straight-skeleton algorithm.

They again maintain a priority queue of edge events and split events. Felkel and Obdržálek [FO99] propose the following procedure in order to determine the split event for a reflex vertex v of P. For each edge e they determine a point p_e on the bisector ray emanated from v which has equal orthogonal distance to e and the incident edges of v. The edge e defines a triangle Δ_e (possibly infinite) that is bounded by e and the bisector rays emanated from the incident vertices of e. If p_e does not lie in Δ_e then they ignore p_e. Among all edges e of P, with p_e lying in Δ_e, they select the one point p_e whose distance from v is smallest. They add a split event for v at this point into our priority queue.

This algorithm takes $O(nr + n \log n)$ time in order to compute the straight skeleton. Unfortunately, the procedure which computes the next split event does not necessarily determine

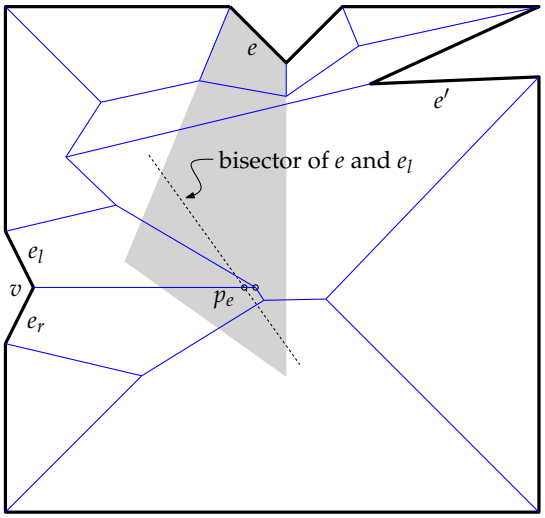


Figure 13: The point p_e does not mark the split event of the reflex vertex v. The vertex v actually causes a crash with the edge e', which happens after v passed p_e.

the correct split event. In Figure 13 we give an example where the split event of a vertex v is not computed correctly. Different issues concerning the correctness are mentioned in Yakersberg [Yak04].

1.4.3 Implementations

To the best of our knowledge the only published runtime statistics are due to Felkel and Obdržálek [FO99], who published the runtime consumption of their code for seven datasets. Beside the implementation of Felkel and Obdržálek, the only implementation available is the straight-skeleton code by Cacciola [Cac04] that is shipped with CGAL [CGA]. The algorithm behind the CGAL implementation was originally based on the algorithm by Felkel and Obdržálek [FO99]. However, Cacciola has significantly adapted the original algorithm in order to get it to work correctly.[23] The current version, CGAL 3.8, can compute the straight skeleton of polygons with holes. Unfortunately, no details on the implemented algorithm are published. In Section 2.5.4, we present experimental results that yield an $O(n^2 \log n)$ runtime performance and an $O(n^2)$ memory footprint for the CGAL implementation.

1.4.4 Summary

The best known lower bounds for the straight skeleton of simple polygons is $\Omega(n)$ resp. $\Omega(n \log n)$ for simple polygons with holes and planar straight-line graphs. This poses a significant gap to the fastest algorithms known so far, see Table 1. The fastest algorithm is due to Eppstein and Erickson [EE99] with a runtime of $O(n^{1+\epsilon} + n^{8/11+\epsilon} r^{9/11+\epsilon})$ for pla-

23 Based on personal e-mail correspondence with the author in 2010.

Algorithm	Time	Space	PSLG	Impl.
[AAAG95]	$O(nr \log n)$	$O(n)$	No	Yes
[AA96]	$O(n^3 \log n)$	$O(n)$	Yes	Yes
[EE99]	$O(n^{1+\epsilon} + n^{8/11+\epsilon} r^{9/11+\epsilon})$	$O(n^{1+\epsilon} + n^{8/11+\epsilon} r^{9/11+\epsilon})$	Yes	No
[CV02]	exp. $O(n \log^2 n + r\sqrt{r} \log r)$		No	No

Table 1: Summary of known straight-skeleton algorithms and their time and space complexities. The input contains n vertices and r denotes the number of reflex wavefront vertices. The column "PSLG" denotes whether the algorithm accepts a planar straight-line graph as input. The column "Impl." denotes whether the algorithm is suitable for implementation from a practical point of view.

Algorithm	Time	Space	Impl.
[EE99]	$O(n^{17/11+\epsilon})$	$O(n^{17/11+\epsilon})$	No
[CV02]	$O(n\sqrt{n} \log n)$	$O(n\sqrt{n})$	No

Table 2: Summary of known motorcycle-graph algorithms and their time and space complexities for n motorcycles. The column "Impl." denotes whether the algorithm is suitable for implementation from a practical point of view.

nar straight-line graphs. For simple non-degenerate polygons Cheng and Vigneron [CV07] presented an algorithm with a slightly better expected runtime of $O(n \log^2 n + r\sqrt{r} \log r)$.

On the implementation side there is the straight-skeleton code by CGAL which accepts polygons with holes as input. It is a wavefront-type algorithm which is roughly based on the algorithms due to [AAAG95, FO99]. The only straight-skeleton algorithm which is suitable for implementation and accepts planar straight-line graphs as input was presented by Aichholzer and Aurenhammer [AA98]. However, no implementation is available and no runtime statistics have been published.

1.5 GENERALIZATIONS AND RELATED PROBLEMS

1.5.1 Linear axis

The linear axis is a variation of the straight skeleton for simple polygons P introduced by Tănase and Veltkamp [TV04a]. The difference between the linear axis and the straight skeleton is a more general definition of the initial wavefront at reflex vertices of P.

The idea is that a reflex vertex v of P does not emanate a single reflex wavefront vertex but k reflex wavefront vertices $v_1, \ldots, v_k$, with $1 \leq k$. The wavefront edges connecting two consecutive vertices v_i, v_{i+1} propagate with unit speed and for each v_i the two incident edges span identical angles, see Figure 14. The skeleton structure which results from this initial wavefront and the ordinary straight-skeleton wavefront propagation is called *linear axis*. For

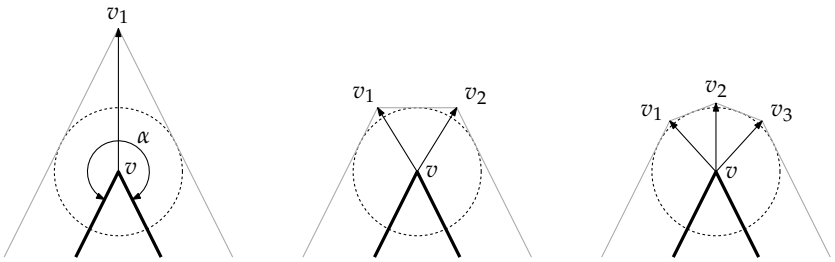


Figure 14: The wavefront (gray) of the linear axis at a reflex vertex v of the input polygon (bold) for different numbers k of emanated wavefront vertices $v_1, \ldots, v_k$, with $k = 1, 2, 3$. The angles between consecutive wavefront edges are equal.

$k = 1$ the linear axis is identical to the straight skeleton. Denoting by $\alpha \in [\pi, 2\pi)$ the interior angle at the vertex v of P, the speeds of the reflex wavefront vertices are given by

$$\frac{1}{\cos \frac{\alpha - \pi}{2k}}. \tag{1.3}$$

Note that this expression tends towards 1 for large k. In particular, the speeds are bounded by $\sqrt{2}$ for $k \geq 2$, whereas for $k = 1$ the reflex wavefront vertices can get arbitrarily fast when α gets close to 2π. The larger k gets, the better the wavefront approximates a circular arc. Tănase and Veltkamp [TV04a] proved that the linear axis converges to the medial axis as k grows for all reflex vertices. The concept of the linear axis has later been generalized to planar straight-line graphs by Vyatkina [Vya09a].

Since the linear axis follows the same wavefront propagation process as the straight skeleton, one can basically apply the algorithms known for straight skeletons in order to compute the linear axis. For $k \in O(1)$ at all reflex vertices, we obtain the same time and space complexities. However, Tănase and Veltkamp showed that if k is chosen large enough such that the medial axis and the linear axis differ in a sufficiently small amount then the linear axis can be obtained from the medial axis in linear time.

Of course, we can further generalize the linear axis by not requiring that consecutive wavefront edges in Figure 14 have identical angles. Basically, we can emanate any polygonal chain as wavefront as long as the wavefront edges have orthogonal distance to v that is equal to the time elapsed so far. We also refer to the discussion in Section 1.2.2 on various types of wavefront polygons emanated at isolated vertices and terminal vertices of the input graph.

The advantages of the linear axis compared to the straight skeleton are all, more or less, related to a reduced speed of the reflex vertices. From a practical point of view, we expect that the linear axis is easier to compute in terms of numerical stability. Tănase and Veltkamp [TV04a] mention that the polygon decomposition application from Section 1.3.5 would also benefit if the speed of the reflex wavefront vertices is limited. Finally, we want to review the mitered offset application for NC-machining, see Section 1.3.1. A tool path based on the straight skeleton can lead to long paths at very sharp reflex vertices. The offset curve based on the linear axis would give us a shorter path while still avoiding a continuous contact of the tool with reflex vertices, as it would be the case for the medial axis.

1.5.2 Weighted straight skeleton

Eppstein and Erickson [EE99] were the first to mention the weighted straight skeleton, where the wavefront edges are allowed to propagate at different speeds. The geometry of the weighted straight skeleton has not yet been systematically investigated and, in particular, its relation to the motorcycle graph is unclear. However, one can observe that basic properties of the unweighted straight skeleton do not carry over to the weighted case. For example, in the unweighted case every straight-skeleton face is monotone w. r. t. its defining wavefront edge, cf. Lemma 2.3 in Section 2.1. As Figure 15 (a) illustrates, this is not necessarily the case for the weighted straight skeleton. Moreover, important theorems concerning alternative characterizations of the straight skeleton do not hold for the weighted case. Eppstein and Erickson [EE99] presented a simple polygon for which Theorem 2.7 does not hold in the weighted case, see Figure 16. The same figure also serves as a counter-example to Theorem 2.11.

Beside the fact that certain properties do not carry over from the unweighted straight skeleton, we observe that the definition of the weighted straight skeleton is tricky in the presence of parallel input segments. Figure 15 shows two wavefront edges e_1, e_2 that propagate with unit speed and e_3 that propagates with speed 2. Let us rotate the edge e_1 around its right endpoint such that e_1 and e_3 become parallel in both subfigures. In the limit we obtain two identical inputs in both subfigures, but in subfigure (a) the straight-skeleton arc between e_1 and e_3 is pointing to the left, whereas in subfigure (b) this arc is pointing to the right. This example poses a singularity in the naive wavefront-based definition of the weighted straight skeleton. Also note that in subfigure (a) the arc between e_1 and e_3 is convex, but the arc encloses an angle greater than $\pi/2$ with e_3. Analogously for subfigure (b), where this arc is reflex but e_3 and the arc encloses an angle less than $\pi/2$. For the unweighted straight skeleton, however, we could characterize reflex and convex arcs by the angle enclosed with its defining wavefront edges.

Lemma 1.11. *Let v denote a wavefront vertex incident to two edges e_a and e_b, where e_a propagates with speed $a > 0$ and e_b propagates with speed $b > 0$. Further assume that e_a and e_b span an angle of $\alpha \in (0, 2\pi)$ at the side where the wavefront propagates to. Then the trajectory of v encloses with e_b an angle α_b and the wavefront vertex v has a speed of $\frac{1}{\sin \alpha_b}$, where*

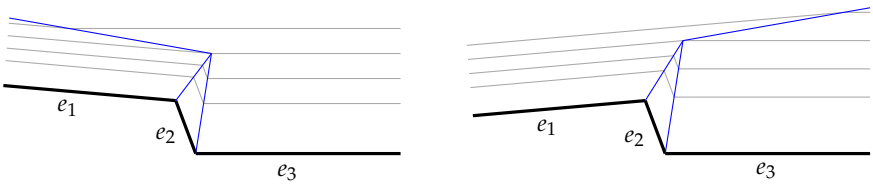
$$\alpha_b = \frac{\pi}{2} - \arctan \frac{\cos \alpha + \frac{a}{b}}{\sin \alpha}. \tag{1.4}$$

Proof. Let us consider Figure 17. Simple trigonometry leads to $x = -a \cdot \sin \alpha$, $y = a \cdot \cos \alpha$, $z = \frac{b+y}{\tan(2\pi-\alpha)}$ and finally to $\alpha_b = \pi/2 + \arctan \frac{x+z}{b}$. □

In the unweighted case, i. e. $a = b$, we obtain $\alpha_b = \alpha/2$ as expected. If we consider a/b fixed but less than 1 we observe the following behavior of α_b at $\alpha = \pi$.

$$\lim_{\alpha \nearrow \pi} \alpha_b = \pi \qquad \lim_{\alpha \searrow \pi} \alpha_b = 0. \tag{1.5}$$

Both equations correspond to the two cases illustrated in Figure 15, with v denoting the wavefront vertex on the arc between e_1 and e_3 and $e_b = e_3$. The left equation is related to Figure 15 (a) while the right equation describes the situation in Figure 15 (b).



(a) (b)

Figure 15: The weighted straight skeleton of three input edges. The edges e_1 and e_2 propagate at unit speed, but e_3 propagates with speed 2. If we rotate e_1 such that e_1 and e_3 become parallel, we obtain two different straight skeletons in the limit for the two subfigures.

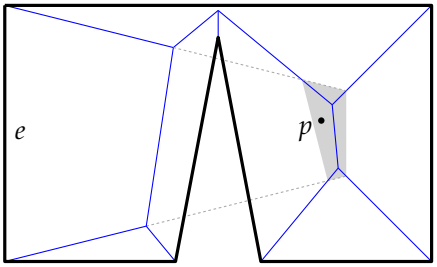


Figure 16: Assume all edges propagate with unit speed, but e has speed 4. Then the edge slab of e cuts off the shaded area from the terrain, i. e. the edge slab reaches below $\mathcal{T}(P)$ at point p. (This figure is based on Figure 6 in [EE99].)

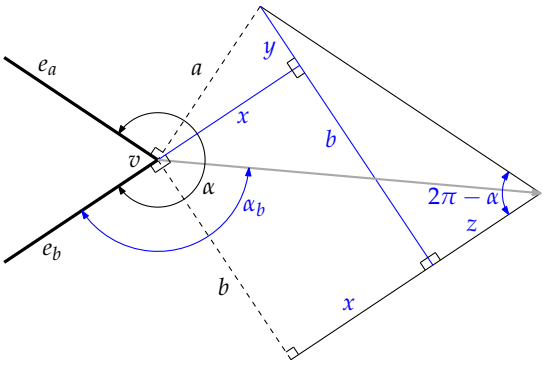


Figure 17: The speed and direction (gray arrow) of a vertex v which is incident to two edges e_a and e_b. The edge e_a is propagating with speed a and e_b is propagating with speed b.

As mentioned in Section 1.4.2.3, Eppstein and Erickson [EE99] presented an algorithm for the weighted straight skeleton of planar straight-line graphs with n vertices which uses $O(n^{8/5+\epsilon})$ time and space.

1.5.3 Straight skeleton of polyhedra in $\mathbb{R}^3$

Straight skeletons of three-dimensional polyhedra have first been mentioned by Demaine et al. [DDLS05], who investigated so-called *hinged dissections* of polyhedra. A dissection of two polyhedra is a tessellation of the first polyhedron into polyhedral pieces such that the second polyhedron can be built by the pieces of the first one. Demaine et al. [DDLS05] discuss so-called hinged dissections which are special dissections where the polyhedral pieces are hinged together at points or edges. It is an open question whether there exists a hinged dissection for every pair of polyhedra. However, the straight skeleton of polyhedra is used by Demaine et al. in order to compute hinged dissections for a certain class of polyhedra.

The straight skeleton $\mathcal{S}(P)$ of a three-dimensional polyhedron P is defined by a wavefront propagation process, where the faces of P move inwards with unit speed in a self-parallel fashion. The set of points which are traced by the intersections of two adjacent faces is called the straight skeleton. Demaine et al. mention that $\mathcal{S}(P)$ gives a decomposition of P into cells such that one cell belongs to exactly one face of P. Further, Demaine et al. cite a personal communication with J. Erickson, in which it is mentioned that the straight skeleton in $\mathbb{R}^3$ is no longer uniquely defined, because at certain points in time one can make multiple decisions to continue the offset propagation.

Barequet et al. [BEGV08] were the first to investigate algorithms for the straight skeleton of polyhedra P of certain types. Firstly, they mention a lower bound of $\Omega(n^2)$ for the size of $\mathcal{S}(P)$ of polyhedra with n vertices. Note that the medial axis and the straight skeleton are identical for a convex polyhedron and the lower bound of $\Omega(n^2)$ due to Held [Hel94] can be applied. Secondly, Barequet et al. mention $O(n^{3+\epsilon})$ as the best known upper bound due to Sharir [Sha94].

Furthermore, Barequet et al. [BEGV08] presented a straight-skeleton algorithm for a certain class of polyhedra. First, they start with polyhedra made of voxels, i. e. axis-parallel cubes of identical size. Second, an extension to polyhedra with axis-parallel edges is presented. In the latter case the complexity of the straight skeleton can be bound to $O(n^2)$. Barequet et al. [BEGV08] present two algorithms for n-vertex polyhedra with axis-parallel edges. The first runs in $O(n^2 \log n)$ time, the second in $O(k \log^{O(1)} n)$ time, where k denotes the size of the straight skeleton. Putting both algorithms together results in a runtime of $O(\min\{n^2 \log n, k \log^{O(1)} n\})$.

An interesting result of Barequet et al. [BEGV08] is that general simple n-vertex polyhedra exist, where the complexity of the straight skeleton is super-quadratic, namely $\Omega(n^2\alpha^2(n))$, where $\alpha(.)$ denotes the extremely slowly growing inverse Ackermann function. Furthermore, Barequet et al. shed more light on the ambiguity issue of straight skeletons of polyhedra.

1.5.4 City Voronoi diagrams

The straight skeleton was used by Aichholzer et al. [AAP04] in order to compute the so-called city Voronoi diagram which is presented as follows. Let C denote a planar straight-line graph with c vertices and edges that are axis-parallel. The graph C models a public transit system like the subway. The problem is to find the shortest path between two vertices x and$y$ in the plane, where the distance measure on $\mathbb{R}^2 \setminus C$ is given by the Manhattan metric. Furthermore, one may enter and leave the public transit at any point of C with zero delay. The public transit is assumed to move on the edges of C with speed $v > 1$.

Aichholzer et al. [AAP04] define by $Q_C(x, y)$ the temporal distance according to the quickest path from x to y. This distance measure Q_C induces a Voronoi diagram on a set S of n vertices, which is called the *city Voronoi diagram* $\mathcal{V}_C(S)$ *of* S *w. r. t.* C. Aichholzer et al. show that $\mathcal{V}_C(S)$ is a subset of the additively and multiplicatively weighted[24] straight skeleton of an input which is based on S and C. This enables Aichholzer et al. [AAP04] to compute the city Voronoi diagram by using straight-skeleton algorithms. Interestingly, Aichholzer et al. were able to show that the input to the straight-skeleton algorithm has sufficiently nice properties such that the framework of abstract Voronoi diagrams of Klein [Kle89] can be applied. Note that for general input it is not possible to interpret the straight skeleton as abstract Voronoi diagram, see Section 2.1. This observation finally leads to an algorithm which uses $O(n \log n + c^2 \log c)$ time and $O(n + c)$ optimal space.

24 We use the term "multiplicatively weighted" if the wavefront edges propagate at different speeds. We use the term "additively weighted" if the some wavefront edges are allowed to start propagating after some time.

2 COMPUTING THE STRAIGHT SKELETON

In order to make straight skeletons applicable in practice we seek an algorithm which is (i) easy to implement and (ii) exhibits an actual runtime that is relatively close to linear in the input size. Fortune started his introduction [For00] to the 27th volume of *Algorithmica* with the following words:

> It is notoriously difficult to obtain a practical implementation of an abstractly described geometric algorithm. This difficulty arises in part from the conceptual complexity of many geometric algorithms, which often use sophisticated data structures or require other complex algorithms as subroutines. The difficulty also arises because many algorithms are designed and described to achieve good asymptotic behavior in the worst case, ignoring behavior in more realistic situations.

In this chapter we present a novel straight-skeleton algorithm which is (i) easy to implement, (ii) exhibits a runtime that is close to linear in practice, (iii) accepts planar straight-line graphs as input, and (iv) has a lower worst-case runtime complexity than the triangulation-based algorithm by Aichholzer and Aurenhammer [AA98].

We start with an investigation of the number of flip events that occur for the algorithm by Aichholzer and Aurenhammer [AA98] in Section 2.2. We first show a few results regarding the gap between $\Omega(n^2)$ and $O(n^3)$ for the worst-case number of flip events and prove that we can exploit Steiner triangulations in order to completely avoid all flip events. This insight motivates an algorithm for straight skeletons of non-degenerate polygons that is based on motorcycle graphs, see Section 2.3. In order to generalize this algorithm to arbitrary planar straight-line graphs we introduce a generalization of the motorcycle graph in Section 2.4 and present an extension of our straight-skeleton algorithm in Section 2.5. Extensive runtime experiments in Section 2.5.4 reveal that our straight-skeleton implementation Bone exhibits an actual runtime consumption of $O(n \log n)$ in practice. Most results in the Sections 2.2 to 2.5 have been presented in the following publications:

- [HH10b] S. Huber and M. Held. Straight Skeletons and their Relation to Triangulations. In *Proc. 26th Europ. Workshop Comput. Geom.*, pages 189–192, Dortmund, Germany, Mar 2010
- [HH10a] S. Huber and M. Held. Computing Straight Skeletons of Planar Straight-Line Graphs Based on Motorcycle Graphs. In *Proc. 22nd Canad. Conf. Comput. Geom. (CCCG 2010)*, pages 187–190, Winnipeg, Canada, Aug 2010
- [HH11c] S. Huber and M. Held. Theoretical and Practical Results on Straight Skeletons of Planar Straight-Line Graphs. In *Proc. 27th Annu. ACM Sympos. Comput. Geom.*, Paris, France, to be published 2011

2.1 GEOMETRIC PROPERTIES OF THE STRAIGHT SKELETON

In this section we build a collection of geometric properties of straight skeletons and their relation to motorcycle graphs. Most of the following geometric properties were presented in [AAAG95, AA98, EE99, CV07]. Throughout this section, we denote by G a planar straight-line graph. It turns out that the terrain $\mathcal{T}(G)$ is often useful to prove various properties of the straight skeleton and its relation to motorcycle graphs. In many cases the proofs become much easier when the scene considered is lifted to $\mathbb{R}^3$. The following lemma is a simple but useful tool for proofs of this kind.

Lemma 2.1. *Consider two distinct points p, q on $\mathcal{T}(G)$. Then the straight line through p and q has a slope of at most* 1*.*

Proof. We project p and q onto the plane $\mathbb{R}^2 \times \{0\}$ and denote the results by p' and q'. The intersection of $\mathcal{T}(G)$ with a vertical curtain above $[p'\,q']$ results in a plane polygonal chain that starts at p and ends at q. All segments of this chain have a slope of at most 1, because each segment results from the intersection of the vertical curtain with a facet of $\mathcal{T}(G)$ and all facets have exactly slope 1. As a consequence, the supporting line $\overline{pq}$ through p and q has also a slope of at most 1. □

Definition 2.2. We denote by $e(t)$ the set of points that are occupied by the wavefront edge e at time t and by $\overline{e(t)}$ we denote the supporting line of $e(t)$. If e is emanated by a terminal vertex or an isolated vertex v of G then we define $\overline{e(0)} := \lim_{t \searrow 0} \overline{e(t)}$.

Note that the propagating wavefront edge e can be split at certain points in time. That is, $e(t)$ can be expressed as the union of straight-line segments. Furthermore, we get that $f(e) = \bigcup_{t \geq 0} e(t)$. For a terminal vertex v we obtain that $\overline{e(0)}$ is equal to the supporting line of v that is perpendicular to the single incident edge of v.

Aichholzer et al. [AAAG95] proved the following lemma for general bisector graphs of simple polygons instead of straight skeletons. These graphs correspond to generalized roofs as discussed in Section 1.2.3. Aichholzer and Aurenhammer [AA96] presented this lemma for planar straight-line graphs G. We rephrase their original proof using Lemma 2.1.

Lemma 2.3 ([AA96])**.** *Let e be a wavefront edge of G. The face $f(e)$ is monotone w. r. t. $\overline{e(0)}$.*

Proof. It has to be shown that an intersection of $f(e)$ with a line perpendicular to $\overline{e(0)}$ is connected. Assume that this is not the case for a line l that is perpendicular to $\overline{e(0)}$. One can find two points x and y on the boundary of $f(e)$ such that the open segment (x, y) is not contained in $f(e)$. We lift the problem to $\mathbb{R}^3$ and put a vertical curtain on $[xy]$, intersect it with $\mathcal{T}(G)$ and obtain a polygonal chain L. Let us denote its endpoints with $\hat{x}$ and $\hat{y}$, which are contained in $\hat{f}(e)$. The supporting line of $\hat{x}, \hat{y}$ has a slope of exactly 1, but note that L is not contained in $\hat{f}(e)$ by assumption. Hence, L contains at least one segment which has a slope that is greater than 1. But this is a contradiction to Lemma 2.1. □

Lemma 2.4 ([AA96])**.** *The straight skeleton $\mathcal{S}(G)$ has exactly $2n + t - 2$ number of nodes, where t denotes the number of terminals in G.*

Similar to Lemma 1.2, the lemma holds if the degree of all nodes is three. In the presence of multi split events the degree of the resulting nodes have to be taken into account accordingly.

The proof of Aichholzer and Aurenhammer is based on the analysis of their algorithm, see Section 1.4.2.2. Recall that their algorithm computes a constrained triangulation of G and then simulates the wavefront by maintaining the triangulation. A topological event of the wavefront corresponds to a decrease of the number of triangles by one. Note that the so-called flip events leave the number of triangles invariant. Therefore, the number of events happening to the wavefront corresponds to the number of triangles collapsed. The number of triangles that remain after the last event happened corresponds to the number of unbounded arcs resp. infinite nodes of $\mathcal{S}(G)$. Aichholzer and Aurenhammer [AA96] argue that the number of initial triangles equals the number of nodes, including the infinite nodes.

Proof. One observes that the initial triangulation tessellates the plane into $2n-2$ triangles. Let us consider an embedding of G onto the three-dimensional sphere, in a local neighborhood of the north pole. We identify the north pole with the origin of the plane and we consider the south pole as an additional vertex which models the locus at infinity. Then an induction-type argument easily shows that any triangulation of $n \geq 3$ vertices plus the infinite vertex comprises $2n-2$ triangles. Finally, each terminal vertex gives rise to an additional triangle in order to take the additional wavefront edges at terminal vertices into account. Summarizing, the initial triangulation comprises $2n-2+t$ triangles. □

THE STRAIGHT SKELETON AS A VORONOI DIAGRAM An obvious yet essential question is whether the straight skeleton can be interpreted as a generalized or abstract Voronoi diagram. The benefit of such a connection is obvious: Voronoi diagrams have been extensively studied in the past decades and there are efficient algorithms for a wide variety of generalizations of Voronoi diagrams [Kle89, KMM93, Yap87, AS95]. Aichholzer and Aurenhammer [AA96] considered the abstract Voronoi framework by Klein [Kle89] in order to investigate this question. The idea of Klein is that one does not consider a distance function in order to define the Voronoi region of each input site, but to define mutual bisector curves between each pair of input sites. Klein [Kle89] gave a set of axioms that need be fulfilled by this set of bisectors, in order to define an abstract Voronoi diagram. The generalized Voronoi diagrams of points, straight-line segments and circular arcs are covered by this framework as well. In addition, generalizations to other distance functions can be represented within this framework. Recently, Klein et al. [KLN09] revisited this framework and presented a conciser set of axioms for the bisector system.

Unfortunately, Aichholzer and Aurenhammer [AA96] pointed out that the framework of Klein cannot be applied to define straight skeletons. Let us consider a bisector between two input sites a, b motivated by the straight skeleton. That is, the bisector is the set of points that are reached at the same time by the wavefronts of a and b. The claim is that the resulting bisector system does not fulfill the axioms of [KLN09]. Following the notation of [KLN09] we denote by $J(a,b)$ the bisector between the site a and b. The bisector tessellates the plane into two halves. The half which belongs to a is denoted by $D(a,b)$ and the other by $D(b,a)$. The Voronoi region $\mathcal{V}(a)$ of a is defined by the intersection $\bigcap_{s \neq a} D(a,s)$ among

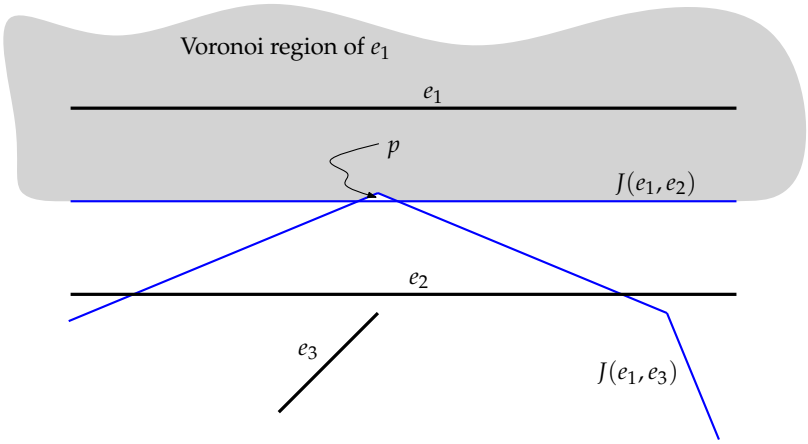


Figure 18: An attempt to define $\mathcal{S}(G)$ using abstract Voronoi diagrams. The bisectors $J(e_1, e_2)$ and $J(e_1, e_3)$ are depicted in blue. The attempt fails because e_3 influences the Voronoi region of e_1. In particular, the point p is not contained in the Voronoi region of e_1.

all input sites s. See Figure 18 for an example. We observe that the Voronoi region of e_1 is influenced by the presence of e_3. However, the straight-skeleton wavefronts of e_3 are blocked by e_2. Note that if we would remove e_2 then $J(e_1, e_3)$ would pose a correct part of the resulting straight skeleton. The basic reason for this strange behavior is that the bisectors motivated by straight skeletons do not fulfill the axioms of Klein et alii. In particular, their axioms include that $D(e_1, e_2) \cap D(e_2, e_3) \subset D(e_1, e_3)$, which is clearly not the case in our example: The point p is contained in the set at the left-hand side but not in the right-hand side. Moreover, p is not contained in any Voronoi region at all.

The concept of abstract Voronoi diagrams by Klein et al. [Kle89, KLN09] distills the very essence of Voronoi diagrams from an abstract point of view. Having this interpretation in mind, straight skeletons are essentially different to Voronoi diagrams, even though they share common properties at the first sight. Nevertheless, Aichholzer and Aurenhammer [AA98] mentioned that for special case of rectilinear polygons P, the Voronoi diagram of P in the L^∞-space and the straight skeleton of P coincide.

THE TERRAIN MODEL AND PIECEWISE-LINEAR FUNCTIONS Aichholzer et al. [AAAG95] investigated whether partially defined linear distance functions could be used in order to obtain an alternative, non-procedural way of defining the straight skeleton $\mathcal{S}(G)$. More precisely, the idea is to define for each wavefront edge e a real-valued linear function $d_e(.)$ on a subset of $\mathbb{R}^2$ such that the lower envelope of the graphs of these functions forms the terrain $\mathcal{T}(G)$. For a point $p \in \mathbb{R}^2$, the value $d_e(p)$ would indicate the time when p is hit by the wavefront edge e. Aichholzer et al. already showed that the domain of d_e cannot only depend on e: In Figure 18, the domain of d_{e_3} must somehow take care for the presence of e_2. We can summarize this insight as follows:

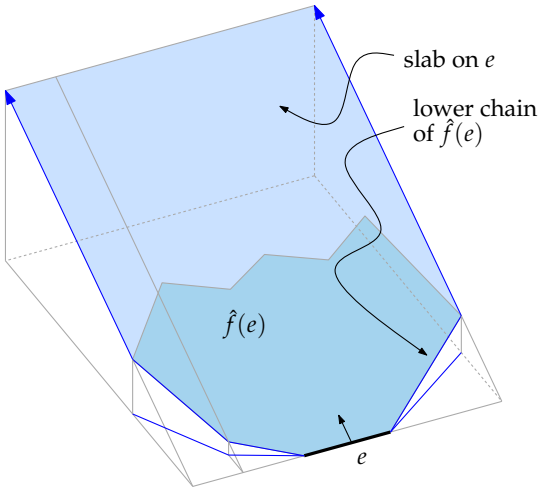


Figure 19: The slabs in $\mathbb{R}^3$ are bounded from below by the lower chain of $\hat{f}(e)$.

Observation 2.5 ([AAAG95])**.** *If one attempts to define a partially linear function for each wavefront edge e of G such that the lower envelope of their graphs is identical to $\mathcal{T}(G)$ then the domain of each function d_e cannot only depend on the defining wavefront edge e.*

Let us consider the face $f(e)$ of a wavefront edge e of G. By Lemma 2.3, the boundary of $f(e)$ consists of two monotonic chains with respect to $\overline{e(0)}$.

Definition 2.6 (lower/upper chain of a face)**.** The *lower chain* and the *upper chain* of $f(e)$ are the two monotonic chains w. r. t. $\overline{e(0)}$ that constitute the boundary of $f(e)$. The lower chain contains $e(0)$.

For the lower chain C of a face $f(e)$, we denote by $\hat{C}$ the projection of C onto $\mathcal{T}(G)$. For each wavefront edge e we define an infinite slab that lies on the supporting plane of $\hat{f}(e)$ and is bounded from below by $\hat{C}$ and by two rays at both endpoints of $\hat{C}$ that are perpendicular to $\overline{e(0)}$, see Figure 19. Eppstein and Erickson [EE99] proved the following theorem.

Theorem 2.7 ([EE99])**.** *The lower envelope of all slabs of the wavefront edges of G is identical to $\mathcal{T}(G)$.*

Proof. We first note that for any wavefront edge e of G, the lifted face $\hat{f}(e)$ is contained in the slab that we defined for e. Hence, it remains to show that the terrain $\mathcal{T}(G)$ is not above the lower envelope of the slabs at any point.

Assume that p is a point on $\mathcal{T}(G)$ that indeed lies above the lower envelope. Hence, we can project p vertically onto the lower envelope and obtain a point p'. Then we can project p' along the steepest descent of the slab, where p' lies on, until we hit a point q. Clearly, q and p are on $\mathcal{T}(G)$. Note that $\overline{qp'}$ has slope 1 and, thereby, $\overline{qp}$ must have slope greater than 1. This is a contradiction to Lemma 2.1. □

In the original work by Eppstein and Erickson [EE99] the theorem above is presented for polygonal input, but it is claimed that it also extends to planar straight-line graphs. Further,

they actually used the following set of slabs: For every edge e they defined an *edge slab*, which is bounded from below by e and by two perpendicular rays at each endpoint of e. For every reflex vertex v of G they defined two *reflex slabs* each of which is bounded from below by the valley incident to v and, for each, two rays that are perpendicular to the two incident edges of v. Strictly speaking, in their original phrasing they did not consider that vertex events could happen, i. e., that a valley of $\mathcal{T}(G)$ is not incident to a vertex of G. Nevertheless, the basic idea of their proof also works in the general case, which is based on a careful consideration of the relative positions of the moving wavefront edges. We presented a proof that is based on Lemma 2.1 instead. Our proof appears to be simpler and immediately applies to arbitrary planar straight-line graphs G.

As Eppstein and Erickson [EE99] mentioned, Theorem 2.7 leads to a nice alternative interpretation of the straight skeleton $\mathcal{S}(G)$, based on a partially defined linear functions. The following corollary formulates this interpretation, where d denotes the infimum distance, that is $d(A, B) := \inf_{x \in A, y \in B} d(x, y)$ for two point sets $A, B \subset \mathbb{R}^2$.

Corollary 2.8. *Let e and e' denote two wavefront edges. For any point $p \in f(e)$, whose orthogonal projection line onto $\overline{e'(0)}$ intersects the lower chain of $f(e')$, holds $d(p, \overline{e(0)}) \leq d(p, \overline{e'(0)})$.*

This result enables us to define a point set that is similar to the *cone of influence* in the theory of Voronoi diagrams, cf. [Hel91]. That is, for each wavefront edge e we define a set $\mathcal{CI}(e)$ by the projection of the slab of e onto the plane. Next, we define a distance function d_e between a wavefront edge e and a point p by

$$d_e(p) := \begin{cases} d(\overline{e(0)}, p) & \text{if } p \in \mathcal{CI}(e) \\ \infty & \text{otherwise.} \end{cases} \tag{2.1}$$

Using these distance functions d_e, we can rephrase the previous corollary to

$$f(e) = \bigcap_{e' \in E} \{p \in \mathbb{R}^2 \ : \ d_e(p) \leq d_{e'}(p)\}, \tag{2.2}$$

where E denotes the set of wavefront edges. At the first sight, this result seems to be an alternative characterization of $\mathcal{S}(G)$. However, we want to remark that the sets $\mathcal{CI}(e)$ depend on the length of the reflex arcs of $\mathcal{S}(G)$. In other words, the representation of the faces via (2.2) does not serve as an alternative characterization of straight skeletons.

Eppstein and Erickson [EE99] remarked that Corollary 2.8 does not pose a contradiction to Observation 2.5: The domain of the distance function d_e depends on the length of certain reflex arcs and, therefore, does not only depend on the wavefront edge e.

Lemma 2.9. *The lower chain of the face $f(e)$ of a wavefront edge e is convex.*

Proof. The lemma asserts that the interior angle at each vertex of the lower chain C of $f(e)$ is at most 180°. The segment $e(0)$ encloses with its incident arcs an angle less than 180°, because these arcs lie on the bisectors of e and adjacent edges of e. Hence, it suffices to show that the interior angle at any straight skeleton node on C is at most 180°. We assume the contrary: Suppose that there is a node v on C whose interior angle is reflex. Without loss of generality, we may assume that v is (i) closest to $e(0)$ and (ii) on the left part of C w. r. t. $e(0)$, see Figure 20.

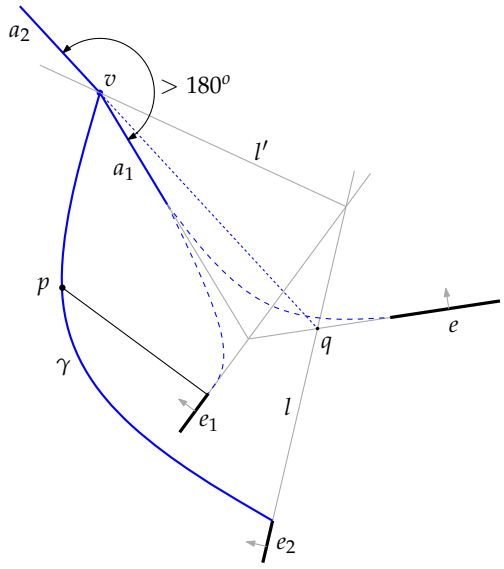


Figure 20: The vertex v on the lower chain of $f(e)$ cannot be reflex.

We denote by a_1 and a_2 the arcs on C that are incident to v such that the sub-chain from a_1 to $e(0)$ is convex. Further, we denote by e_1 and e_2 the other two wavefront edges whose faces contain the arcs a_1 and a_2, respectively. The supporting line of a_2 intersects $\overline{e(0)}$ at a point q. There exists a supporting line l of q such that a_2 lies on the bisector of l and $\overline{e}$. Hence, the edge e_2 lies on l, by construction. Further, we note that $e_2(0)$ lies behind $e(0)$ and also behind $e_1(0)$ w. r. t. the corresponding propagation direction.

Since e_2 reaches v after some time, the lower chain of $f(e_2)$ contains a polygonal chain γ that connects $e_2(0)$ and v. We note that γ is monotone w. r. t. e_2 by Lemma 2.3. Furthermore, γ is monotone w. r. t. the propagation direction of e_2 because the points on γ are swept by the propagating edge e_2. The curve γ contains a point p that can be projected orthogonally onto $e_1(0)$. Corollary 2.8 implies that $d(p, \overline{e_2(0)}) \leq d(p, \overline{e_1(0)})$ because $p \in f(e_2)$. Let us denote by l' the bisector between e_1 and $_2$. That is, l' consists of all points $\overline{e_1(t)} \cap \overline{e_2(t)}$, with $t \geq 0$, and all points left to l' are first reached e_1. However, since p is left to l' it follows that $d(p, \overline{e_1(0)}) < d(p, \overline{e_2(0)})$, which is a contradiction. □

Lemma 2.10. *The lower chain of a face $f(e)$ consists of $e(0)$ and reflex straight-skeleton arcs.*

The following theorem was proved by Cheng and Vigneron [CV07]. It establishes an essential connection between the motorcycle graph and the straight skeleton.

Theorem 2.11 ([CV07])**.** *Let P denote a simple non-degenerate polygon P. The reflex arcs of $\mathcal{S}(P)$ are covered by $\mathcal{M}(P)$.*

Proof. We slightly rephrase the proof in [CV07] in order to fit our setting. First, we declare that $\mathcal{S}(P)$ and $\mathcal{M}(P)$ are restricted to the interior of the polygon P rather than being defined on the whole plane.

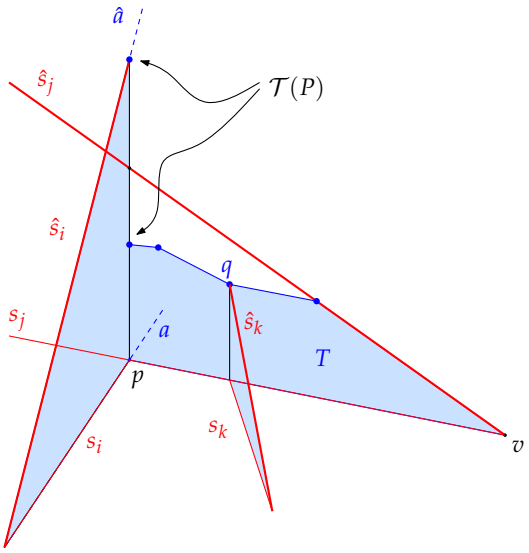


Figure 21: Reflex arcs are covered by motorcycle traces. The set T turns out to be convex which leads to a contradiction concerning the height of $\mathcal{T}(P)$ above p.

Cheng and Vigneron lift the problem to $\mathbb{R}^3$ by means of the terrain model. Let $m_1, \ldots, m_r$ denote the motorcycles from $\mathcal{M}(P)$, where r denotes the number of reflex vertices of P. Further, let s_i denote the trace of the motorcycle m_i and let $\hat{s}_i$ denote the lifted trace of $\hat{m}_i$ by interpreting the third spatial dimension as the time. Recall that each motorcycle m_i starts at a reflex wavefront vertex v of $\mathcal{W}(P, 0)$ and m_i and v have the same velocity. Hence, the valley $\hat{a}$ that belongs to v has the same inclination as the lifted motorcycle trace $\hat{m}_i$.

Now assume that the statement is false and there is indeed a reflex arc a of $\mathcal{S}(P)$ that is only partially covered by the motorcycle trace s_i. Since a continues on the track of m_i, the motorcycle m_i crashed into another motorcycle, say m_j. Let p denote the intersection point of s_i and s_j and let t^* denote the height of $\mathcal{T}(P)$ above p. Among all reflex arcs that are only partially covered, a is chosen such that t^* is lowest. Hence, up to the height t^*, all valleys of $\mathcal{T}(P)$ are covered by tilted motorcycle traces.

The trace s_j starts at a reflex vertex v of P and contains p. Let T denote the intersection of a vertical curtain that is put on the segment $[vp]$ with the set of points below $\mathcal{T}(P)$, see Figure 21. The claim is that T is convex. Assume, to the contrary, that T contains a reflex vertex on its top chain. Let q denote such a reflex vertex that is closest to v. The vertex q cannot be at height zero, which would mean that m_j crashed into a wall — that is, an edge of P — at q. Note that T and $\hat{s}_j$ start with the same inclination, which means that q is below or just at the same height as $\hat{s}_j$. Because q is a reflex vertex of T, there exists a valley at q, and because each valley is covered by a motorcycle trace until height t^*, there is a tilted motorcycle trace $\hat{s}_k$ at q. So either s_j crashed into s_k or the valleys corresponding to m_j and m_k meet at q. Both cases are a contradiction to the initial assumptions. It follows that T is convex and hence below $\hat{s}_j$.

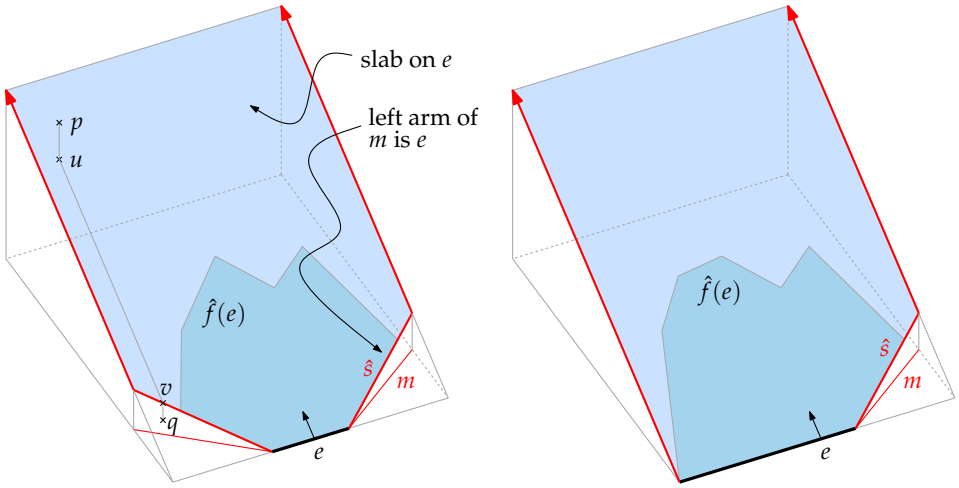


Figure 22: The slab at e is bounded from below by e and the tilted motorcycle traces of motorcycles that have e as an arm. Left: two motorcycles have e as an arm. Right: Only one motorcycle has e as an arm.

The height of $\mathcal{T}(P)$ at p is once given by the valley $\hat{a}$, on one hand, and by the height of T at p, on the other hand. But this is a contradiction, because $\hat{a}$ is strictly above $\hat{s}_j$, which itself is above T. □

The following corollary is a byproduct of the previous proof.

Corollary 2.12 ([CV07])**.** *Let P denote a simple non-degenerate polygon P. The tilted traces $\hat{s}$ of $\mathcal{M}(P)$ are above or just at the same height as $\mathcal{T}(P)$.*

Let us revisit the slab construction from Theorem 2.7. Cheng and Vigneron [CV07] presented a different slab construction scheme that is based on the motorcycle graph. For the matter of consistency, we reformulate their construction as follows.

Let P denote a simple non-degenerate polygon. Each motorcycle m in $\mathcal{M}(P)$ starts from a reflex wavefront vertex v in $\mathcal{W}(P, 0)$, which is incident to two wavefront edges. Recall that we call the one wavefront edge that is left to the track of m the left arm of m and the other wavefront edge the right arm of m. To each wavefront edge e, there can exist one motorcycle whose right arm is e and one motorcycle whose left arm is e, see Figure 22.

Definition 2.13 (lower envelope)**.** Let $\hat{C}$ denote the union of $e(0)$ and the tilted traces of the motorcycles that have e as an arm. For each wavefront edge e we define a slab that lies on the supporting plane of $\hat{f}(e)$ and is bounded from below by $\hat{C}$ and two rays at the endpoints of $\hat{C}$ that are perpendicular to $\overline{e(0)}$. The *lower envelope* of these slabs is denoted by $L(P)$.

Lemma 2.14 ([CV07])**.** *For a simple non-degenerate polygon P holds $L(P) = \mathcal{T}(P)$.*

Proof. Consider a point p on $\mathcal{T}(P)$. Because p sits on a tilted face $\hat{f}(e)$ of a wavefront edge, it follows that p is also contained in the corresponding slab of and, thereby, not below $L(P)$. It

remains to show that a point p of $\mathcal{T}(P)$ is not above $L(P)$. Assume, to the contrary, that p is above any slab of an wavefront edge e, see Figure 22. One can project p down onto the slab and obtain the point u. Then one can project u along the steepest descent of the slab until the lower boundary of the slab is hit at the point v. By Corollary 2.12, the point v is above or just on $\mathcal{T}(P)$. Hence, one can project v onto $\mathcal{T}(P)$ and obtain q, which is below v or identical to v. The slope of the line $\overline{pq}$ is at most 1, by Lemma 2.1. On the other hand, the slope of $\overline{pq}$ is strictly greater than the slope of $\overline{uv}$, which is exactly 1. This is a contradiction. □

Cheng and Vigneron [CV07] pursued the approach of Eppstein and Erickson to define their set of slabs. They reuse the term *edge slab* from [EE99] and, instead of reflex slabs, they introduce the so-called *motorcycle slabs*. The approach presented above, however, extends more naturally to arbitrary planar straight-line graphs, see Section 2.4.

Let us revisit the discussion concerning an alternative characterization of straight skeletons after Corollary 2.8. Recall that the lower envelope of the slabs of Eppstein and Erickson [EE99] does not lead to an alternative characterization of straight skeletons, because the sizes of the slabs depend on the length of reflex arcs. The slabs of Cheng and Vigneron [CV07], however, do indeed lead to an alternative characterization of straight skeletons: Their slabs only depend on the motorcycle graph. Again, a single slab is not only locally defined by the corresponding wavefront edge. Nevertheless, the contributions of Cheng and Vigneron [CV07] only apply to non-degenerate polygons, whereas the results of Eppstein and Erickson [EE99] apply to arbitrary planar straight-line graphs.

We want to give three reasons why Theorem 2.11 appears to be important. Firstly, the motorcycle graph is an extraction of the "essential sub problem" of computing straight skeletons. Hence, it is important to know the geometric relation between straight skeletons and motorcycle graphs. Secondly, the motorcycle graph helps to devise algorithms to compute the straight skeleton. Thirdly, the motorcycle graph helps to devise an alternative characterization of straight skeletons. Aichholzer and Aurenhammer [AA96] already remarked that it is desirable to find a non-procedural definition of straight skeletons. For this reasons we consider our generalization of Theorem 2.11 to arbitrary planar straight-line graphs in Section 2.4 as important.

Cheng and Vigneron [CV07] presented a randomized straight-skeleton algorithm for simple non-degenerate polygons P, which is based on Theorem 2.11, see Section 1.4.2.4. They also present an extension of their algorithm to non-degenerate polygons P with holes. It is easy to see that Theorem 2.11 holds for these slightly more general polygons as well, after generalizing the definition of the motorcycle graph induced by P accordingly.

2.2 THE TRIANGULATION-BASED APPROACH

In this section, we study the number of flip events that occur in the triangulation-based straight-skeleton algorithm by Aichholzer and Aurenhammer [AA98]. The upper bound of $O(n^3)$ is easy to see, cf. Section 1.4.2.2. However, to the best of our knowledge, no n-vertex polygon or planar straight-line graph is known that exceeds a quadratic number of flip events. This circumstance constitutes a gap by a linear factor and the question remains

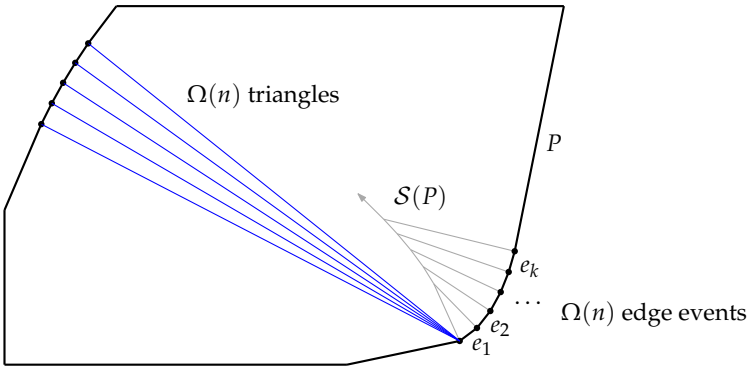


Figure 23: A convex polygon P with n vertices that causes a $\Theta(n^2 \log n)$ runtime for the triangulation-base straight-skeleton algorithm. Triangulation diagonals are shown in blue. Part of the straight skeleton is depicted in gray. The edge events for the edges $e_1, \ldots, e_k$, with $k \in \Omega(n)$, occur in the order $e_1, \ldots, e_k$. For each edge event, the collapsing times of $\Omega(n)$ incident triangles have to be updated, which consumes $\Theta(n^2 \log n)$ time in total.

open, whether the number of flip events is actually bound by $O(n^2)$. Besides, note that processing edge and split events already takes $O(n^2 \log n)$ time: Even though there are only $\Theta(n)$ edge and split events in total, a single event requires up to $O(n \log n)$ time, because the collapsing times of $O(n)$ triangles may need an update in the priority queue. In fact, even a modest convex polygon can lead to a runtime consumption of $\Theta(n^2 \log n)$, as illustrated in Figure 23.

2.2.1 The number of reappearances of diagonals

In Section 1.4.2.2, we explained the $O(n^3)$ bound for the number of flip events, by observing that three constantly moving points do not get collinear more than twice and every flip event corresponds to a collinearity of a triple of vertices. But note that not every collinearity does necessarily correspond to a flip event. Let us consider a triangulation diagonal between two vertices A and B. This diagonal may disappear because another vertex S crosses this diagonal during the propagation process. However, it could be possible that the diagonal is restored by subsequent flip events as illustrated in Figure 24. In order to have the diagonal AB restored, the vertex S has to back off such that A and B see each other again, as illustrated in Step 2. Hence, the vertices A, B and S got collinear twice and S cannot cause a flip event for AB again. In other words, S cannot flip the diagonal AB twice. Nevertheless, Figure 24 does not prove that the diagonal AB can be actually restored. We just considered the necessary topological changes and not the proper geometric setting that leads to these topological changes. How often can a single diagonal, say AB, actually reappear? If this number would be in $O(1)$ then we would conclude that the number of flip events is in fact in $O(n^2)$, because there are only $\binom{n}{2}$ pairs of vertices that can be connected by a diagonal. Unfortunately, the following lemma does not even state that a single diagonal can reappear $\Omega(n)$ times, but $\Omega(n)$ diagonals can each reappear $\Omega(n)$ times, which leads to $\Omega(n^2)$ flip events in total.

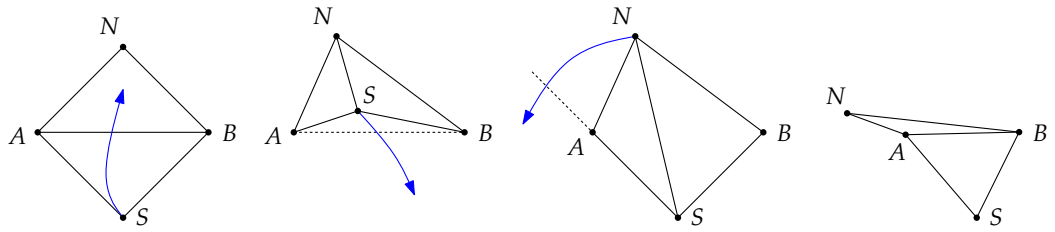


Figure 24: A sequence of flip events which leads to the reappearance of the diagonal AB between the vertices A and B.

Lemma 2.15 ([HH10b]). *There exist polygons P with n vertices and triangulations T of P such that $\Omega(n)$ diagonals each reappear $\Omega(n)$ times during the wavefront propagation.*

Proof. We prove the statement in three steps. In each step we give a geometric setting of moving vertices, for which a specific sequence of flip events occurs. At first, we describe how a single diagonal AB reappears twice. In the second step, we extend the construction such that AB reappears $\Omega(n)$ times. In the third step, we show how k diagonals $A_1B, \ldots, A_kB$ reappear each $\Omega(n)$ times, with $k \in \Omega(n)$.

Let us consider six vertices A, B, S_0, S_1, N_0, N_1. The goal is to carefully construct start points and velocities for these six vertices such that the topological transitions, as illustrated in Figure 25, are executed. We denote by $V(t)$ the position of a vertex V at time t. The basic building block of this proof works as follows: We want that the vertex S_0 flips the diagonal AB and after that backs off such that the vertices A and B see each other again. All vertices are bound to move with constant speed. We let A and B drive vertically upwards and we assume that $A(0)$ is at the origin and $B(0)$ at $(1, 0)$. We denote by v_A and v_B the speeds of A and B and we further assume that $v_A = 2$ and $1 < v_B/v_A \ll 2$.

The movement of the vertex S_0 is completely determined by choosing the loci of $S_0(1)$ and $S_0(1+\Delta)$, where $\Delta \in (0, 1/4)$ is fixed. The locus of $S_0(1)$ is chosen to be on the straight-line segment $[A(1)B(1)]$ and $S_0(1+\Delta)$ is chosen to be on $[A(1+\Delta)B(1+\Delta)]$. We further require that S_0 moves to the north-east, i. e. that $S_0(1+\Delta)$ lies strictly to the right to $S_0(1)$. The determinant $\det(A(t), B(t), S_0(t))$, which is

$$\begin{vmatrix} A_x(t) & B_x(t) & S_{0,x}(t) \\ A_y(t) & B_y(t) & S_{0,y}(t) \\ 1 & 1 & 1 \end{vmatrix}, \tag{2.3}$$

is a quadratic polynomial in t and its sign corresponds to the orientation of the three vertices. (The x- and y-coordinates of the vertices A, B, S_0 are denoted by subscripts.) We observe that for some $t > 1 + \Delta$ the points $A(t), B(t), S_0(t)$ are in clockwise position. This is because the vertex S_0 moves to north-east and $v_B/v_A > 1$. Knowing that $\det(A(t), B(t), S_0(t))$ is a quadratic polynomial with roots $\{1, 1+\Delta\}$, and by observing that $A(t), B(t), S_0(t)$ are in clockwise position for some $t > 1 + \Delta$, we conclude that the triangle $A(t), B(t), S_0(t)$ is clockwise for all $t < 1$, counter-clockwise for all $t \in (1, 1+\Delta)$ and clockwise again for all $t > 1 + \Delta$. In other words, we constructed a vertex S_0 which indeed executes the first two transitions in Figure 25.

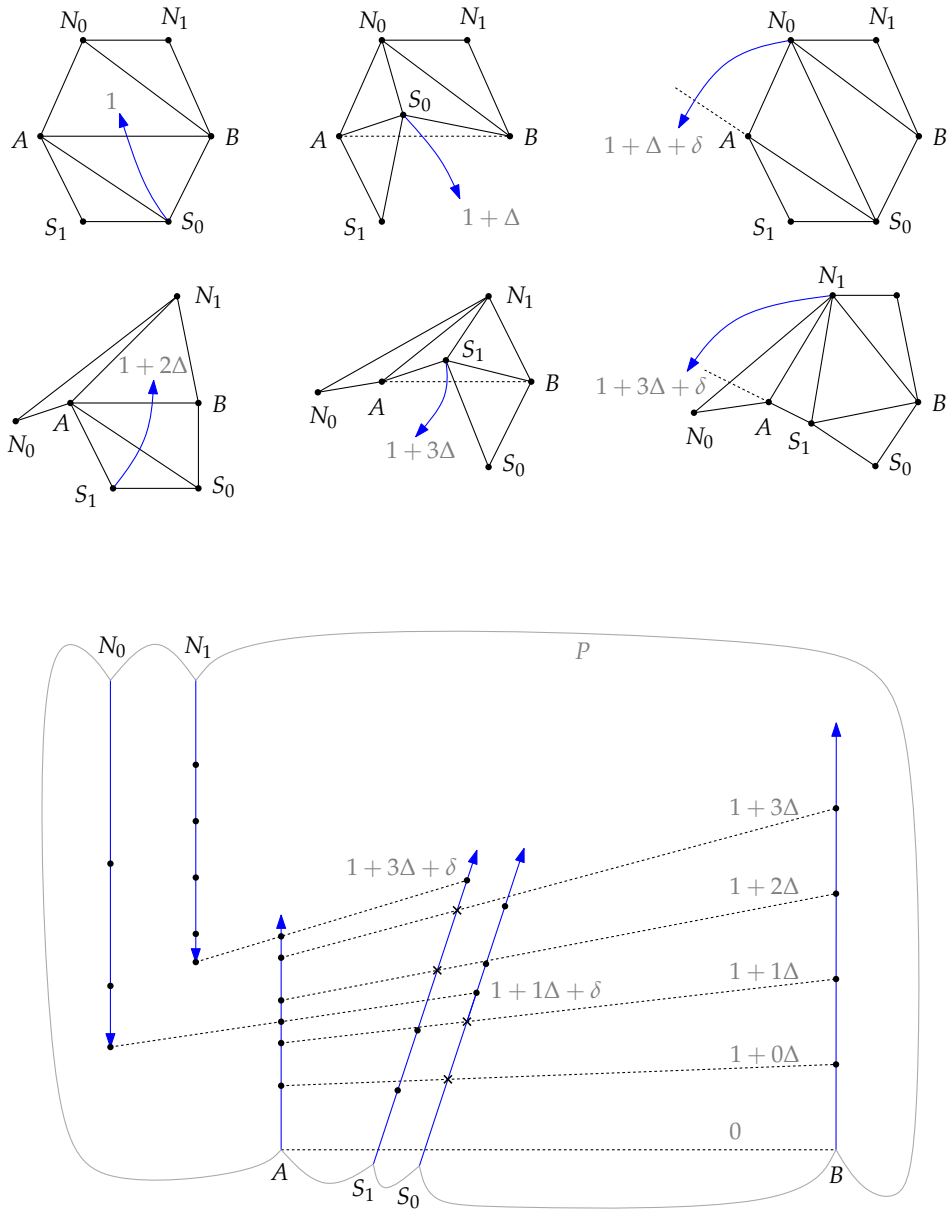


Figure 25: Step 1 of the proof of Lemma 2.15: The diagonal AB reappears twice since the geometric setting at the bottom produces the sequence of flip events at the top. Bottom: Every black dot denotes the position of a vertex at the time given which is depicted in gray. A cross mark indicates collinearity for A and B with S_0 resp. S_1. Top: Each figure illustrates a topological transition. The second reappearance is caused by the sixth transition.

According to our desired sequence of transitions, we need a vertex N_0 such that $N_0(t)$ gets collinear with $A(t)$ and $S_0(t)$ for some $t > 1 + \Delta$, say at $t = 1 + \Delta + \delta$, where $\delta \in (0, \Delta/2)$ is fixed. (In the sequel, we have to choose δ small enough.) We place a vertex N_0 that moves southwards and parallel to A. Furthermore, we choose $N_{0,y}(0) = 8$ such that $N_0(0)$ is strictly above $B(1 + 4\Delta)$. Next, we request that $N_0(1 + \Delta + \delta)$ is on the supporting line $\overline{A(1 + \Delta + \delta)\, B(1 + \Delta + \delta)}$. Hence, at time $1 + \Delta + \delta$ the vertex N_0 causes the recreation of the diagonal AB by flipping the diagonal N_0S_0.

In order to conclude the first step of the proof, we repeat the life-cycle of the diagonal AB, by using the vertices S_1, N_1 instead of S_0, N_0. We place the vertex S_1 between A and S_0 and let S_1 move parallel to S_0. The locus of $S_1(t)$ is chosen to be on $[A(t)B(t)]$ for $t \in \{1 + 2\Delta, 1 + 3\Delta\}$. Applying the same argument as for S_0, we observe that the triangle $A(t), B(t), S_1(t)$ is clockwise for all $t < 1 + 2\Delta$, counter-clockwise for all $t \in (1 + 2\Delta, 1 + 3\Delta)$ and clockwise again for all $t > 1 + 3\Delta$. Note that S_0 is already behind $\overline{A(t)B(t)}$ for $t > 1 + \Delta$. Hence, S_1 causes a flip event for the diagonal AB at $t = 1 + 2\Delta$ and falls back at $t = 1 + 3\Delta$ such that the vertices A and B see each other again for $t > 1 + 3\Delta$. Also note that the introduction of S_1 does not interfere with the transitions caused by S_0 and N_0, since S_1 is behind the supporting line of A, B, S_0 at $t = 1 + \Delta$. Hence, by choosing δ small enough, S_1 is also behind the supporting line of A and S_0 at $t = 1 + \Delta + \delta$.

Finally, we place a vertex N_1 between N_0 and A which again moves southwards and parallel to A. We assume that N_0 and N_1 start from the same horizontal line and we require that $N_1(1 + 3\Delta + \delta)$ is collinear with $A(1 + 3\Delta + \delta)$ and $S_1(1 + 3\Delta + \delta)$. Hence, the diagonal AB is recreated again by an edge flip of the diagonal N_1S_1.

To sum up, we are able to construct a geometric setting of moving vertices such that the diagonal between the vertices A and B reappears twice. However, in order to finish Step 1 of the proof, we have to guarantee that there is a polygon P and a triangulation T of P that realizes the sequence of transitions in Figure 25. First, we note that we can choose Δ very small such that the speeds of N_0 and N_1 get arbitrarily close. Next, we note that if we choose $v_B/v_A > 1$ but arbitrarily close to 1 then the points $S_1(0)$ and $S_2(0)$ approach the supporting line of $A(0)$ and $B(0)$. Hence, we can construct a polygon P, as in Figure 25, such that only the vertices A, S_1, S_0, B, N_1, N_0 of P are reflex and connected by convex chains. By choosing the incident polygon edges of each of these vertices accordingly, we obtain the desired velocities for the propagating wavefront vertices. The initial triangulation T of P contains the edges given in Figure 25 and the remaining faces can be triangulated arbitrarily.

For Step 2 we extend our construction by adding vertices $S_2, \ldots, S_m$ from right to left between S_1 and A, with $m \in \Omega(n)$. We repeat the construction scheme presented above for these new vertices. Furthermore, we choose $\Delta < 1/2m+2$ in order to make the new vertices fit. Likewise we add vertices $N_2, \ldots, N_m$ from left to right between N_1 and A. In general, for each $i \in \{0, \ldots, m\}$ the vertex S_i causes a flip event for the diagonal AB at time $1 + 2i\Delta$ and falls back behind $\overline{A(t)\, B(t)}$ at time $t = 1 + (2i + 1)\Delta$. At time $t = 1 + (2i + 1)\Delta + \delta$ the vertex N_i gets collinear with $A(t)$ and $S_i(t)$ and causes the recreation of the diagonal AB.

The basic idea of Step 3 is to rename the vertex A to A_1 and to carefully place copies $A_2, \ldots, A_k$ of A_1 from right to left. In the following, we only consider a single reappearance cycle for each diagonal $A_1B, \ldots, A_kB$, caused by flip events due to vertex S_i and the subsequent restoration due to vertex N_i, for an arbitrary $i \in \{0, \ldots, m\}$. We illustrated the topo-

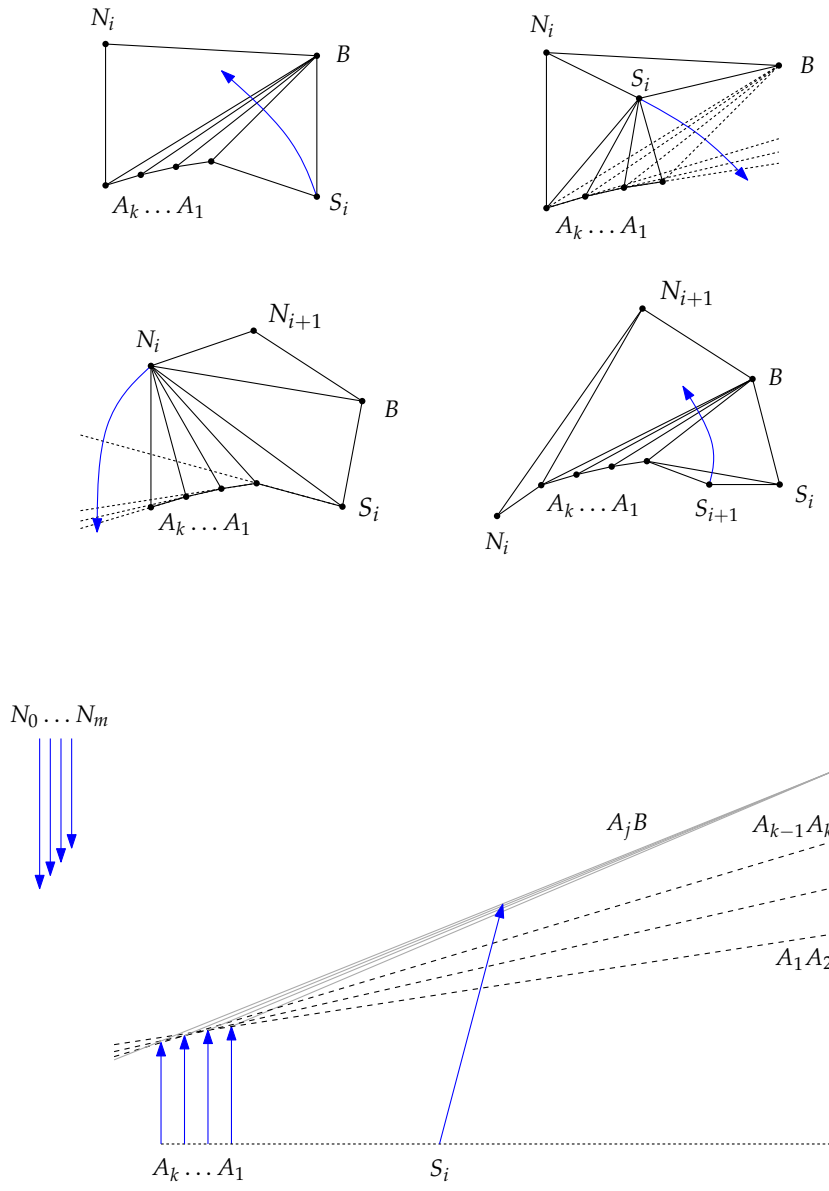


Figure 26: Step 3 of the proof of Lemma 2.15: We place copies of A_1 in order to obtain diagonals $A_1B, \dots, A_kB$. By arranging the vertices $A_2, \dots, A_k$ almost collinear with A_1B the diagonals $A_2B, \dots, A_kB$ behave like the diagonal A_1B from Step 2.

logical transitions in Figure 26. Let us assume that the points $B(0), A_1(0), \ldots, A_k(0)$ are arranged on a horizontal line, see Figure 26. Furthermore, assume that $B(1), A_1(1), \ldots, A_k(1)$ are almost on a straight line, by arranging $A_2(1), \ldots, A_k(1)$ accordingly. However, for any $j \in \{1, \ldots, k-1\}$ each $A_{j+1}(1)$ shall properly see the vertex $B(1)$ and every supporting line $\overline{A_j(1)A_{j+1}(1)}$ shall intersect the trajectory of B in an ascending order. Moreover, the intersection points shall be below but arbitrarily close to $B(1)$, see Figure 26. The idea is to arrange $A_2(1), \ldots, A_k(1)$ very close to $\overline{A_1(1)\,B(1)}$ such that the diagonals $A_1B, \ldots, A_kB$ behave almost the same as the single diagonal AB from Step 2. (Note that if $B, A_1, \ldots, A_k$ are collinear at time 1 then they are collinear at all times, due to the intercept theorems.)

We recall from Step 2 that $\det(A_1(t), B(t), S_i(t))$ is a quadratic polynomial in t with roots $\{1 + 2i\Delta, 1 + (2i+1)\Delta\}$. Furthermore, $A_1(t), B(t), S_i(t)$ is in clockwise position for all $t < 1 + 2i\Delta$, in counter-clockwise position for $t \in (1 + 2i\Delta, 1 + (2i+1)\Delta)$ and again in clockwise position for $t > 1 + (2i+1)\Delta$. Consider $\tau \in (0, \delta)$ to be fixed. It follows that the triangle $A_1(t)B(t)S_i(t)$ is in a strict counter-clockwise position for $t = 1 + 2i\Delta + \tau$. Recall that S_i crosses the diagonal A_1B at $t = 1 + 2i\Delta$. We now choose $B(1), A_1(1), \ldots, A_k(1)$ as close to collinear as possible such that S_i also crosses the diagonal A_jB before $t = 1 + 2i\Delta + \tau$ for all $j \in \{1, \ldots, k\}$. That is, S_i causes a flip event for each diagonal $A_1B, \ldots, A_kB$ within a temporal tolerance of τ after $t = 1 + 2i\Delta$. Hence, the first topological transition in Figure 26 happened after $t = 1 + 2i\Delta + \tau$.

According to the second transition in Figure 26, we need S_i to fall back behind the former diagonals $A_1B, \ldots, A_kB$. However, in addition we require that S_i also falls behind the supporting lines of $\overline{A_j\,A_{j+1}}$ for all $j \in \{1, \ldots, k-1\}$. This circumstance leads to flip events of the diagonals $A_2S_i, \ldots, A_{k-1}S_i$ such that all $A_1, \ldots A_k$ are connected to N_i. From Step 2 we know that S_i falls behind A_1B at time $t = 1 + (2i+1)\Delta$. For Step 3 we choose $B(1), A_1(1), \ldots, A_k(1)$ as close to collinear as necessary such that S_i falls behind the supporting lines $\overline{A_j\,A_{j+1}}$ until $t = 1 + (2i+1)\Delta + \tau$.

In order to execute the third transition in Figure 26, we finally require that N_i crosses the supporting lines $\overline{A_j\,A_{j+1}}$ until $t = 1 + (2i+1)\Delta + \delta + \tau$. After that, all diagonals $A_1B, \ldots, A_kB$ are restored and we completed the reappearance cycle caused by S_i and N_i.

To sum up, we can arrange $A_2(1), \ldots, A_k(1)$ as close as necessary to $\overline{A_1(1)\,B(1)}$ such that the topological transitions in Figure 26 are executed for all S_i, N_i, with $0 \leq i \leq m$, within a temporal tolerance of $\tau \in (0, \delta)$. Furthermore, we observe that at time zero the vertices $B, A_1, \ldots, A_k$ are on a horizontal line. In order to enable diagonals $A_1B, \ldots, A_kB$ in the initial triangulation, we start our simulation at time $-\epsilon$, for a sufficiently small $\epsilon > 0$. □

2.2.2 Good triangulations and bad polygons

As a byproduct of Lemma 2.15, we obtain that polygons P with n vertices and corresponding triangulations exist for which $\Omega(n^2)$ flip events occur. Besides the actual shape of the polygon, this result also hinges on the specific initial triangulation. If the initial triangulation from Lemma 2.15 would not contain the diagonals $A_1B, \ldots A_kB$, but, for example, the diagonals $S_0N_m, \ldots, S_mN_m$ then we would easily avoid the occurrence of $\Omega(n^2)$ flip events. Can we always find for a polygon P a "good" initial triangulation, for which only a few

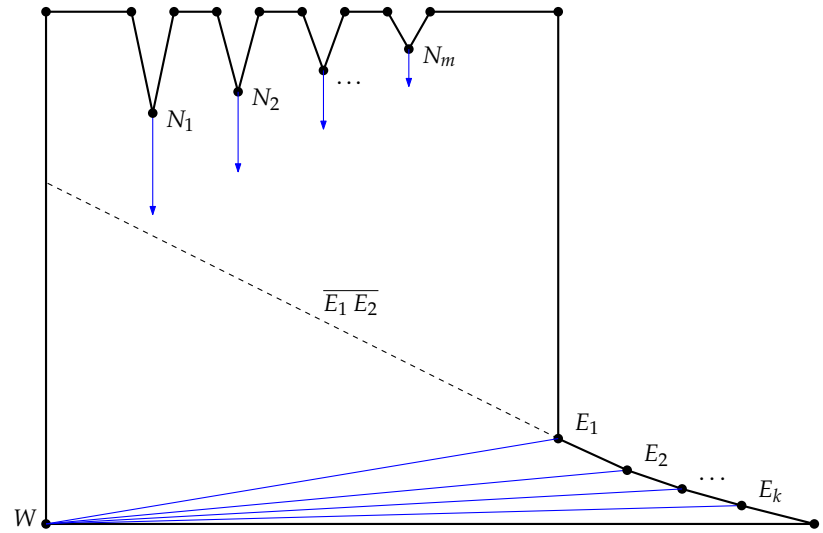


Figure 27: An n-vertex polygon for which any triangulation leads to $\Omega(n^2)$ flip events.

number of flip events occur — say, at most $o(n^2)$ or even $O(n)$? Or, conversely, are there "bad" polygons, for which any triangulation ends up with a large number of flip events?

Lemma 2.16 ([HH10b]). *There exist polygons P with n vertices, for which any triangulation leads to $\Omega(n^2)$ flip events.*

Proof. Let us consider the polygon illustrated in Figure 27. The polygon has $k \in \Omega(n)$ vertices $E_1, \ldots, E_k$ that form a reflex chain. The supporting line $\overline{E_1\,E_2}$ is chosen in a way such that $E_2, \ldots, E_k$ only see the vertex W (except for their neighboring vertices, of course). Hence, any triangulation for P necessarily contains the diagonals $E_1W, \ldots, E_kW$. The idea is that each of these $\Omega(n)$ diagonals is flipped $\Omega(n)$ times by vertices $N_1, \ldots, N_m$, with $m \in \Omega(n)$. We choose the vertex N_i fast enough such that it leads to a split event with the bottom polygon edge before N_{i+1} crosses $\overline{E_1\,E_2}$. Furthermore, we make the vertices $N_1, \ldots, N_m$ fast enough such that the split events for $N_1, \ldots, N_m$ happen before any other split or edge event happens.

If we choose the speeds of $N_1, \ldots, N_m$ accordingly then we observe that N_1 leads to $\Omega(n)$ flip events with $E_1W, \ldots, E_kW$ and we obtain diagonals $N_1E_1, \ldots N_1, E_k$. Next, the vertex N_2 flips the diagonals $N_1E_1, \ldots, N_1E_k$ and we obtain diagonals $N_2E_1, \ldots, N_2E_k$. (In the meanwhile N_1 could have caused a split event already. However, the corresponding diagonals remain incident to a wavefront vertex emanated from this split event and will be flipped by N_2.) At the end, each vertex $N_1, \ldots, N_m$ will flip $\Omega(n)$ diagonals and we obtain $\Omega(n^2)$ flip events in total. □

In order to reduce the number of flip events it seems reasonable to try to re-triangulate the wavefront $\mathcal{W}(P, t)$ at favorable moments. In other words, we could try to pay the costs for computing a new triangulating in exchange to the costs caused for a certain number of flip events. However, we want to remark that one re-triangulation of the polygon constructed in

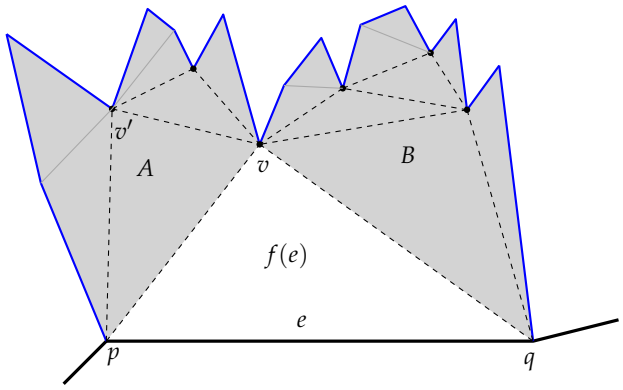


Figure 28: A recursive triangulation scheme for the face $f(e)$ which is free of flip events.

the proof of Lemma 2.16 at any point in time saves at most $O(n)$ flip events: If N_i is already below $\overline{E_1 E_2}$ then all vertices $N_1, \ldots, N_{i-1}$ already caused $\Omega((i-1)n)$ flip events and N_{i+1} is still above $\overline{E_1 E_2}$. Hence, we avoid at most $O(n)$ flip events that would be caused by N_i itself and pay the costs for a re-triangulation.

2.2.3 Steiner triangulations without flip events

Lemma 2.16 tells us that ordinary triangulations do not allow a strategy to keep the number of flip events below $O(n^2)$ for arbitrary polygons. However, we observe that if we use Steiner[1] points, we could reduce the number of flip events significantly for the polygon illustrated in Figure 27. Nevertheless, introducing Steiner points to the initial triangulation in the algorithm by Aichholzer and Aurenhammer [AA98] obtrudes several questions: (i) how do Steiner points interact with the wavefront, (ii) how do we determine the loci of the Steiner points and (iii) how effective is the introduction of Steiner points in order to reduce the number of flip events? Regarding the first question, we assume that the Steiner points keep still. Furthermore, it seems natural to maintain the property that the triangulation keeps the area $\mathbb{R}^2 \setminus \bigcup_{t' \leq t} \mathcal{W}(P, t')$ triangulated for any time $t \geq 0$. We give an answer to the remaining two questions by the following lemma.

Lemma 2.17 ([HH10b])**.** *Every simple polygon P with n vertices admits a triangulation that employs at most $n-2$ Steiner points and is free of flip events.*

Proof. We prove the statement by first presenting a proper set of Steiner points and a proper triangulation. In the sequel we show that no flip events occur for that particular triangulation.

First, we consider the straight skeleton $\mathcal{S}(P)$ of P, which has $n-2$ inner nodes. Each inner node serves as a Steiner point in our triangulation. Next, we add the arcs of $\mathcal{S}(P)$ as diagonals to the triangulation. It remains to properly triangulate the faces of $\mathcal{S}(P)$.

1 Steiner points are additional points at favorable loci and which can be used for the triangulation.

Let $f(e)$ denote an arbitrary face of $\mathcal{S}(P)$ for a wavefront edge e of P. We triangulate $f(e)$ in a recursive manner as follows. Let us denote by p and q the endpoints of e. If we consider $f(e)$ as a polygon we know due to Lemma 2.3 that $f(e)$ is monotone and due to Lemma 2.9 that reflex vertices only appear in the upper chain of $f(e)$. If $f(e)$ is convex then we triangulate it arbitrarily. Otherwise we denote by v a reflex vertex of $f(e)$ with minimum orthogonal distance to $\bar{e}$, see Figure 28. We note that the line segments $[pv]$ and $[qv]$ are completely contained in $f(e)$. Otherwise there would be another reflex vertex of $f(e)$ whose orthogonal distance to e would be less than for v. We insert pv and qv as diagonals to the triangulation. In the sequel we call pv and qv the two diagonals that belong to v.

The triangle pqv tessellates $f(e)$ into (at most) two parts. We denote by A the part to which the diagonal pv belongs to and by B the part to which qv belongs. In both parts A and B we proceed recursively: If the part A is convex we triangulate A arbitrarily. Otherwise we denote by v' the reflex vertex of A with minimum orthogonal distance to $\bar{e}$. We add pv' and vv' as diagonals. The triangle pvv' tessellates A into (at most) two parts and so on.

It remains to show that the triangulation presented above is free of flip events. First of all, we observe that during the wavefront propagation all vertices of the wavefront move on the diagonals of the triangulation. Hence, no reflex wavefront vertex can cause a flip event. However, we also have to guarantee that no triangulation diagonal is moving over a Steiner point. More precisely, we have to exclude the case that a triangulation diagonal moves over a Steiner point that appears as a reflex vertex of the polygon $f(e)$. However, if we consider the propagating wavefront within $f(e)$ then we only need to consider the wavefront edge e, which is sweeping $f(e)$ in a self-parallel manner and is split at any reflex vertex of $f(e)$. If there would exist a diagonal that moves over a reflex vertex v of $f(e)$ then it would follow that this diagonal crosses at least one of the two diagonals that belong to v. □

The proof of the above Lemma obviously does not provide a new straight-skeleton algorithm since we employ the straight skeleton in order to construct an appropriate Steiner triangulation. Nevertheless, the lemma tells us at least that there exist triangulations that are free of flip events. Consequently, we can ask for Steiner triangulations which are free of flip events and which are constructed without the help of straight skeletons. We also want to remark that Lemma 2.17 is not restricted to polygons only and we can easily apply the very same proof to planar straight-line graphs as well.

Corollary 2.18. *Every planar straight-line graph G with n vertices admits a triangulation that employs $O(n)$ Steiner points and is free of flip events.*

2.3 A NOVEL WAVEFRONT-TYPE APPROACH

2.3.1 Motivation

In the previous section we learned that the use of Steiner vertices can be advantageous in order to reduce the number of flip events in the triangulation-based straight-skeleton algorithm by Aichholzer and Aurenhammer [AA98]. However, the following question remained open: How do we find a good set of Steiner points without knowing the straight skeleton?

In the original approach a flip event occurs when a reflex wavefront vertex crosses a triangulation diagonal. These flip events are avoided for the Steiner triangulation presented in the proof of Lemma 2.17, because the reflex wavefront vertices move *along* triangulation diagonals. The idea is to find a set of Steiner points and a triangulation such that the trajectories of the reflex wavefront vertices are covered by triangulation diagonals.

Assume we are given a non-degenerate polygon P. (Recall, P is non-degenerate if no two motorcycles of $\mathcal{M}(P)$ crash simultaneously at the same point.) By Theorem 2.11 we know that the motorcycle graph $\mathcal{M}(P)$ covers the reflex arcs of the straight skeleton $\mathcal{S}(P)$. We consider the endpoints of the motorcycle traces of $\mathcal{M}(P)$ as Steiner points and the traces as diagonals of the triangulation. Theorem 2.11 guarantees that a reflex wavefront vertex does not move beyond its corresponding motorcycle trace. Hence, we obtained a similar situation as in Lemma 2.17, where no flip events are caused by reflex wavefront vertices. In the following we will discuss these two questions: (i) what happens when the wavefront meets a Steiner point and (ii) how do we triangulate the remaining faces?

2.3.2 The extended wavefront and a novel straight-skeleton algorithm

Consider a motorcycle trace that starts at a reflex wavefront vertex v of $\mathcal{W}(P,0)$ and ends at an edge of P at point p. By our construction above, we obtain a Steiner point at p and a triangulation diagonal covering the motorcycle trace that starts at v and ends in p. We define that while the wavefront propagates, the Steiner point at p accompanies the moving intersection of $\mathcal{W}(P,t)$ and the motorcycle trace. Metaphorically speaking, the point p "surfs" on the wavefront, along the motorcycle trace, towards v. Analogously, if a motorcycle trace starts at v and crashes into another motorcycle trace at point p then the Steiner point at p keeps still until the wavefront $\mathcal{W}(P,t)$ meets p. From that moment, p starts moving on the motorcycle trace towards v. To sum up, we maintain the intersection of the motorcycle graph $\mathcal{M}(P)$ and the propagating wavefront $\mathcal{W}(P,t)$ and we refer to the additional points as Steiner vertices. More precisely, we call a Steiner vertex, which has not yet been met by the wavefront, a *resting Steiner vertex* and we call a Steiner vertex, which already moves on the intersection of a motorcycle trace and the wavefront, a *moving Steiner vertex*. Every resting Steiner vertex lies on the intersection of two motorcycle traces and eventually becomes a moving Steiner vertex when it is reached by the wavefront, see Figure 29.

Definition 2.19 (extended wavefront)**.** Let P be a simple non-degenerate polygon. We denote by $\mathcal{M}(P,t)$ those parts of $\mathcal{M}(P)$ which have not been swept by $\mathcal{W}(P,t')$ for $t' < t$. The *extended wavefront* $\mathcal{W}^*(P,t)$ is defined as the overlay of $\mathcal{W}(P,t)$ and $\mathcal{M}(P,t)$ by splitting the edges of $\mathcal{W}(P,t)$ at the intersection points accordingly.

Figure 29 illustrates the extended wavefront $\mathcal{W}^*(P,t)$ for a simple polygon P. In the following, we want to remark that P denotes the filled polygon and not only its boundary.

Lemma 2.20. *We denote by P a simple non-degenerate polygon. Let p be a point in the relative interior of $\mathcal{M}(P)$. Then a local disk around p is tessellated into convex slices by $\mathcal{M}(P)$.*

Proof. Since P is non-degenerate it follows that p is also in the relative interior of a motorcycle trace. □

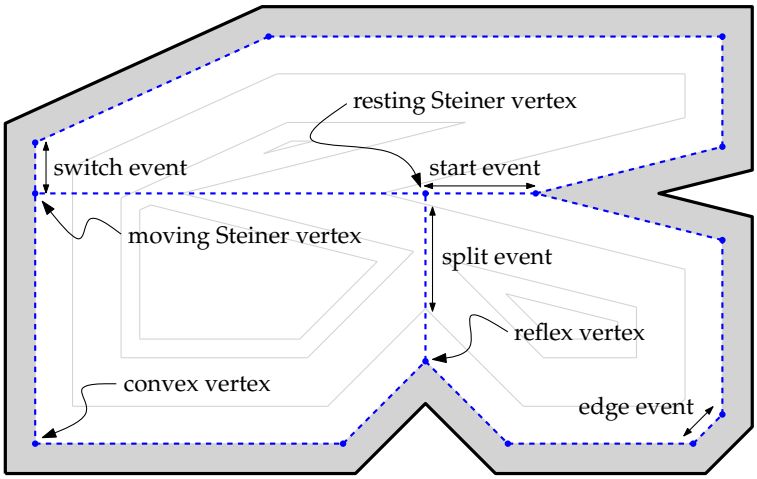


Figure 29: A non-degenerate polygon (bold) and the extended wavefront (dashed) after some time. The area already swept by the wavefront is shaded. The offset curves for three further points in time have been depicted in gray.

Lemma 2.21. *For any $t \geq 0$ the set $P \setminus \bigcup_{t' \in [0,t]} \mathcal{W}^*(P, t')$ consists of open convex faces only.*

Proof. Lemma 2.20 and the fact that at each reflex vertex of P the interior angle is halved by a motorcycle trace, proves the lemma. □

We now tackle the remaining question concerning the triangulation of the faces induced by the extended wavefront. Recall that we regard the nodes of the motorcycle graph as Steiner vertices and the edges of the motorcycle graph as triangulation diagonals. It follows by Lemma 2.21 that during the propagation of the extended wavefront $\mathcal{W}^*(P, t)$ only neighboring vertices on $\mathcal{W}^*(P, t)$ can meet, since the resulting faces are at any time convex. Hence, we can simply determine all topological changes that occur to $\mathcal{W}^*(P, t)$ by just considering neighboring vertices. In other words, there is no need for a triangulation in order to identify the topological changes.

Summarizing, our approach to compute straight skeletons of simple non-degenerate polygons is not to simulate the wavefront $\mathcal{W}(P, t)$, but to simulate the extended wavefront $\mathcal{W}^*(P, t)$. As mentioned in Section 1.4.2.1, the difficult problem when simulating the original wavefront $\mathcal{W}(P, t)$ is the determination of the split events. In our approach, however, split events are easily determined: A moving Steiner vertex meets a reflex wavefront vertex and both move on the same motorcycle trace. Moreover, every topological change in the extended wavefront $\mathcal{W}^*(P, t)$ is indicated by the collision of two neighboring vertices. In other words, each topological change corresponds to an edge of $\mathcal{W}^*(P, t)$ that collapses to zero length and vice versa. Our algorithm maintains a priority queue Q that contains for each topological change an event. We fetch all events in chronological order and process them. That means, we maintain the extended wavefront depending on the type of event that happened. We distinguish the following types of events. (A detailed discussion is given in

Section 2.5.) Note that the different types of events are easily distinguished by the types of vertices involved.

- **(Classical) edge event**: Two convex vertices u and v meet. We merge u and v to a single convex vertex and recompute the collapsing times of the two incident edges of $\mathcal{W}^*(P,t)$. Further, we add the straight-skeleton arcs that have been traced out by u and v to the growing straight-skeleton structure. As a special case, we check whether a whole triangle of $\mathcal{W}^*(P,t)$ collapsed by this edge event.
- **(Classical) split event**: A reflex vertex u and a moving Steiner vertex v meet at point p and both move against each other. We add the straight-skeleton arc that has been traced out by u. In general, $\mathcal{W}^*(P,t)$ is split into two parts. We create corresponding convex vertices that start at p connect them with the resulting parts.
- **Start event**: A reflex vertex or a moving Steiner vertex u meets a resting Steiner vertex v. The vertex v becomes a moving Steiner vertex and slides along one incident edge of u. Technically, we have to split an incident edge of u accordingly.
- **Switch event**: A convex vertex u meets a moving Steiner vertex or a reflex Steiner vertex v. The convex vertex u migrates from one convex face of $\mathcal{W}^*(P,t)$ to a neighboring one, by jumping over v. Technically, the vertex v splits the opposite edge of u and we have to recompute the speed of v.
- All remaining combinations of vertices meeting are guaranteed not to happen. For instance, a convex vertex will never meet a resting Steiner vertex, because a resting Steiner vertex can only have reflex vertices or Steiner vertices as neighbors.

The correctness of the algorithm follows directly from Lemma 2.21 and Theorem 2.11. The latter guarantees that reflex vertices stay within their corresponding motorcycle traces, i. e. they lead to a split event before they would move beyond the extended wavefront.

2.3.3 Runtime analysis and conclusion

Each event involves the recomputation of the collapsing times for a certain amount of edges of the extended wavefront and subsequent modifications in the priority queue Q. Note that each vertex of the extended wavefront has degree two or three. Hence, every event involves only a constant amount of modifications and, thereby, can be handled in $O(\log n)$ time.

We know that we have $O(n)$ edge events and $O(r)$ split and start events in total, where $r \in O(n)$ denotes the number of reflex vertices of P. The number of switch events is bound by $O(nr)$, because a convex vertex can meet a moving Steiner vertex only once. Note that the size of the extended wavefront is linear and the priority queue Q contains at most as many events as the number of edges of the extended wavefront. Hence, our algorithm runs in $O(n)$ space.

Lemma 2.22. *Let P denote a simple non-degenerate polygon with n vertices, where r vertices are reflex. If the motorcycle graph $\mathcal{M}(P)$ is known then our algorithm runs in $O((n+k)\log n)$ time and $O(n)$ space, where $k \in O(nr) \subset O(n^2)$ denotes the number of switch events that occurred.*

For real-world input, it seems very unlikely that $\Omega(n^2)$ switch events actually occur. This would mean that $\Omega(n)$ moving Steiner vertices actually meet with $\Omega(n)$ convex vertices.

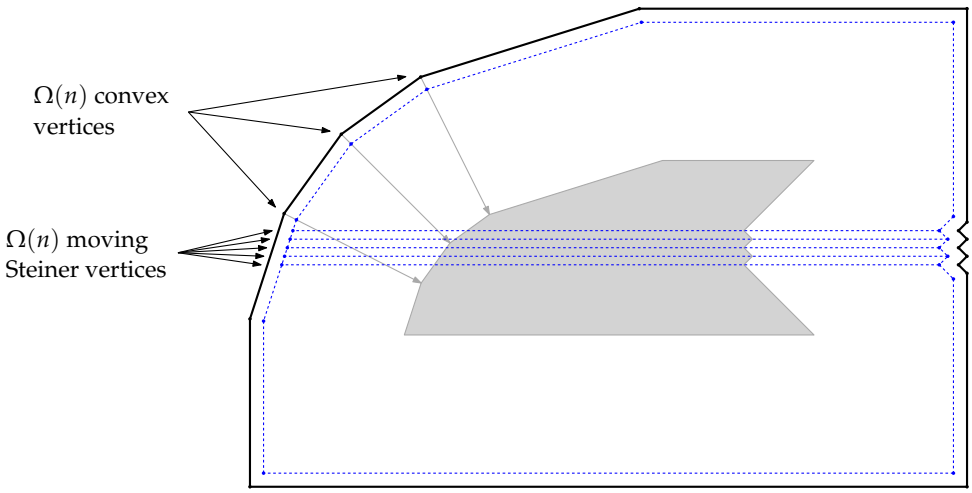


Figure 30: A polygon P for which $\Omega(n^2)$ switch events occur.

This observation is also confirmed by extensive runtime tests in Section 2.5.4. However, a worst-case example, for which $\Theta(n^2)$ switch events actually occur, can be constructed, see Figure 30. Note that we have to take care that the traces of the motorcycles are almost parallel within the whole polygon. Furthermore, we require that the trajectories of convex vertices do not intersect before the $\Omega(n^2)$ switch events occur. That is, the trajectories more or less approach the same point. In other words, this worst-case example is highly contrived.

The motorcycle graph $\mathcal{M}(P)$ of a polygon P with r reflex vertices can be computed in $O(r^{17/11+\epsilon})$ time and space in theory, using the algorithm by Eppstein and Erickson [EE99], cf. Section 1.4.2.3, and in $O(r\sqrt{r}\log r)$ time, using the algorithm by Cheng and Vigneron [CV07], cf. Section 1.4.2.4. In practice, the motorcycle graph can be computed in $O(r^2 \log r)$ time by a straight-forward algorithm enhanced with a priority queue. In Section 3.3, we will present our motorcycle graph implementation Moca, which exhibits an $O(r \log r)$ runtime for practical input.

Summarizing, the algorithm presented in this section has a worst-case runtime complexity of $O(n^2 \log n)$, while it is still easy to implement. This constitutes an improvement by a linear factor compared to the $O(n^3 \log n)$ worst-case complexity of the algorithm by Aichholzer and Aurenhammer [AA98]. However, their algorithm is capable of computing the straight skeleton of planar straight-line graphs instead of non-degenerate polygons.

In the following sections we will extend our approach to arbitrary planar straight-line graphs. In the first place this means that we get rid of the non-degeneracy assumption. In order to achieve this we will need to generalize the motorcycle graph accordingly. As a byproduct, this will also result in an alternative characterization of straight skeletons of planar straight-line graphs. Furthermore, the generalized motorcycle graph will allow us to extend the algorithmic approach to planar straight-line graphs.

2.4 A GENERALIZED MOTORCYCLE GRAPH

2.4.1 Motivation and definition

In order to extend the algorithm that was presented in the previous section to arbitrary planar straight-line graphs, we have to generalize the motorcycle graph from simple non-degenerate polygons to arbitrary planar straight-line graphs. While a generalization to non-degenerate planar straight-line graph appears to be straight-forward, the main challenge is to eliminate the need for the assumption of Cheng and Vigneron [CV07]. That is, we need to allow that two motorcycles may crash simultaneously into each other. No matter how the actual generalization is finally realized, we demand the following two properties to hold for the motorcycle graph $\mathcal{M}(G)$ of a planar straight-line graph G:

1. The motorcycle graph $\mathcal{M}(G)$ has to cover the reflex arcs of $\mathcal{S}(G)$ and
2. the overlay of the motorcycle graph $\mathcal{M}(G)$ and G has to yield a convex tessellation of the plane.

Following the definition of the motorcycle graph induced by a simple polygon in Section 1.2.4, we have to specify the motorcycles and the walls that are induced by a given planar straight-line graph G. First of all, we consider the edges of G to be walls. Secondly, we shoot for each reflex wavefront vertex v of $\mathcal{W}(G,0)$ a motorcycle that starts at v and has the same velocity as v. Note that if G forms a polygon then we obtain motorcycles inside and outside of the polygon. Let us consider a planar straight-line graph G, for which a vertex event occurs at point p. This implies that at least two motorcycles m and m' crashed simultaneously into each other at p. Since a new reflex straight-skeleton arc emanates at p, we need to start a new motorcycle which is going to cover this new straight-skeleton arc. Hence, it will be necessary to generalize the concept of motorcycle graphs such that new motorcycles can emerge at certain moments in time. As a consequence, our generalized motorcycles are specified by a start point, a velocity and, in addition, a start time.

For the matter of simplicity we will rephrase the procedure, by which we obtain the set of motorcycles induced by a planar straight-line graph G. Let us denote by $e(t)$ the straight-line segment occupied by a wavefront edge e at time t. We demand that two wavefront edges e and e' of G belong to each motorcycle m and the position of m at time t is given by the intersection $\overline{e(t)} \cap \overline{e'(t)}$. We call the wavefront edge left to the trace of m the *left arm of* m and the other wavefront edge the *right arm of* m. Obviously, by specifying the start point and the two arms of a motorcycle m, we implicitly specified the start time of m, too.

For any reflex wavefront vertex v in $\mathcal{W}(G,0)$ we define a motorcycle m that starts at v and whose arms are the two incident wavefront edges of v, see Figure 31 (a). Hence, the motorcycle m has the same velocity as v by construction. Furthermore, we note that each terminal vertex of G gives rise to two motorcycles by definition, see Figure 31 (b). If two or more motorcycles $m_1, \ldots, m_k$ crash simultaneously at a point p then we consider a local disk D around p. This disk is tessellated into slices by the motorcycle traces. If all slices are convex then we ignore this event. In particular, if another motorcycle reached p earlier, the disk is tessellated into convex slices. In the opposite case, there is exactly one non-convex slice. We may assume that (i) the traces of $m_1, \ldots, m_k$ appear in counter-clockwise order at

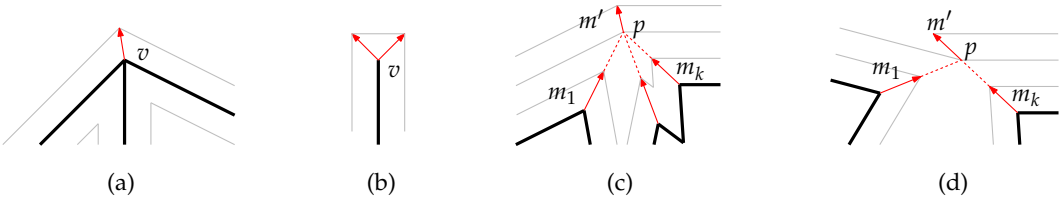


Figure 31: Four different situations for which motorcycles are launched. (a) each reflex wavefront vertex at time zero gives rise to one motorcycle. (b) a terminal vertex gives rise to two motorcycles. (c,d) if two or more motorcycles crash at p such that the traces tessellate a local disc into non-convex slices then we launch a new motorcycle.

p and (ii) the non-convex slice is bounded by the traces of m_1 and m_k. We distinguish the following two cases:

1. The left arm of m_1 and the right arm of m_k span a reflex angle (at the side which has not yet been swept by the wavefront at this time). Then we launch a new motorcycle m', which starts at p and whose arms are given by the left arm of m_1 and the right arm of m_k, see Figure 31 (c). This case is necessary in order to cover reflex straight-skeleton arcs that emanate from a vertex event.
2. The left arm of m_1 and the right arm of m_k span a convex angle. Then we shoot a new motorcycle m' which continues the movement of m_k. That is, m' starts at p and has the same pair of arms as m_k, see Figure 31 (d).

   Letting the motorcycle m' continue the movement of m_k appears to be somewhat arbitrary at the first sight. However, by this definition, we obtain the nice property that the arms of a motorcycle always span a reflex angle at the propagation side. This property, however, turns out to be useful in the proof of Theorem 2.26.

   Also note that Lemma 2.25 becomes trivial for the case illustrated in Figure 31 (d). Indeed, Lemma 2.25 would be also trivial in the case that is illustrated in Figure 31 (c) if we would launch an additional motorcycle that continues the movement of m_k. However, it is unclear whether Lemma 2.24 would remain true.

We note that we do not have to simulate the wavefront in order to apply the above case distinction: it suffices to know the propagation direction of the arms of the motorcycles. Furthermore, it could happen that an arm of a motorcycle already vanished in the wavefront propagation process while the motorcycle still moves. The terminology of *arms* turns out to be advantageous in the subsequent analysis.

In both cases of the previous case distinction, we call $m_1, \ldots, m_k$ the ancestors of m'. In particular, we call m_k the right-most ancestor and m_1 the left-most ancestor. Note that m_1 resp. m_k can again have ancestors. We define the right-most ancestor chain of m' as the union of the trace of m', the trace of m_k, the trace of the right-most ancestor of m_k and so on. The left-most ancestor chain is defined likewise.

Definition 2.23 (motorcycle graph induced by a planar straight-line graph)**.** Let G denote a planar straight-line graph. We consider the edges of G as walls and consider the set of motorcycles as elaborated above. The *motorcycle graph* $\mathcal{M}(G)$ *induced by* G is defined as the arrangement of the resulting motorcycle traces.

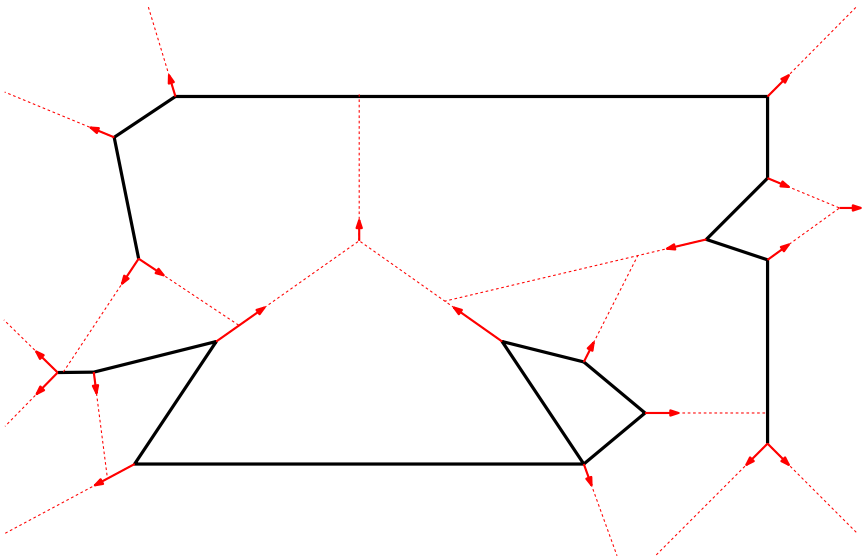


Figure 32: The motorcycle graph $\mathcal{M}(G)$, depicted in red, which is induced by the planar straight-line graph G (bold).

In Figure 32 we have illustrated the motorcycle graph $\mathcal{M}(G)$ for a planar straight-line graph G. Figure 3 on page 9 illustrates the corresponding straight skeleton $\mathcal{S}(G)$.

2.4.2 Geometric properties of the generalized motorcycle graph

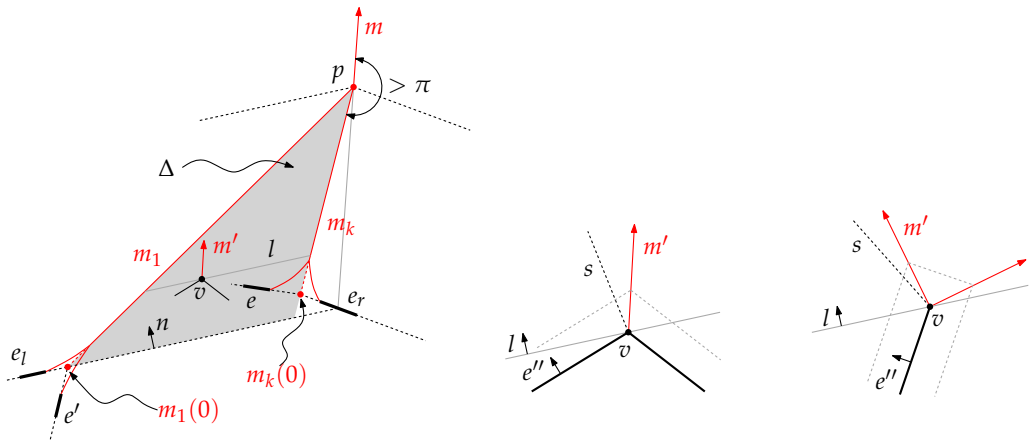
Lemma 2.24. *The motorcycle graph $\mathcal{M}(G)$ of a planar straight-line graph G contains $O(n)$ motorcycle traces.*

Proof. Let $r \in O(n)$ denote the number of reflex wavefront vertices in $\mathcal{W}(G, 0)$. Each motorcycle that does not start from a vertex of G starts from the crash point of at least two other motorcycles. Hence, we obtain that $\mathcal{M}(G)$ comprises at most $2r - 1$ motorcycles traces. □

Lemma 2.25 ([HH11c])**.** *Let p denote a point in the relative interior of $\mathcal{M}(G)$. A local disk around p is tessellated into convex slices by the traces of $\mathcal{M}(G)$.*

Proof. If p is in the relative interior of a trace, the lemma is trivially true. Let us assume that p is the endpoint of $k \geq 2$ motorcycle traces. We denote the motorcycles that crashed at p by $m_1, \ldots, m_k$. Consider a local disk D around p. If D is tessellated into convex slices by the traces of $m_1, \ldots, m_k$ then our assertion holds again. Assume that there is a non-convex slice. We may assume that the motorcycles are numbered in a cyclic order such that (i) their corresponding traces appear counter-clockwise around p and (ii) the trace of m_1 and m_k bound the reflex slice.

If the left arm of m_1 and the right arm of m_k span a convex angle then there is a motorcycle m which continues the movement of m_k and the assertion holds again. So, assume that the



(a) (b) (c)

Figure 33: Convex tessellation at simultaneous crashes. (a) if m and m_k span a reflex angle then the launch of m is impossible. (b–c) a close-up of the vertex v, where v is (b) a non-terminal vertex and (c) a terminal vertex.

left arm of m_1 and the right arm of m_k span a reflex angle. Hence, there is a motorcycle m which started from p and shares its left arm with m_1 and its right arm with m_k. We have to prove that (i) the traces of m_1 and m span a convex angle and (ii) the traces of m_k and m span a convex angle. Without loss of generality, we assume that m and m_k span an angle greater than π. (The other case is symmetric.) The basic idea is to prove that under this assumption, the existence of m is contradicted because (i) m_1 or m_k crashes before reaching p or (ii) m_1 and m_k reach p after another motorcycle was at p before.

We denote by e_l and e_r the left and right arm of m, respectively. We note that m shares its right arm with m_k and its left arm with m_1. Further, we denote by e' the right arm of m_1 and by e the left arm of m_k, see Figure 33 (a). Neither m_1 nor m_k need to start at time zero. We denote by R the right-most ancestor chain of m_1, which contains the trace of m_1 and ends at an endpoint of $e'(0)$. (Recall that we denote by $e(t)$ the straight-line segment occupied by e at time t.) Likewise, we denote by L the left-most ancestor chain of motorcycle traces of m_k.

First we show that $e(0)$ is above $\overline{e_l(0)}$, i. e., $e(0)$ is on the propagation side of $\overline{e_l(0)}$: The point $m_1(0)$ must lie on $\overline{e_l(0)}$. (If m_1 does not start at time zero, we interpret $m_1(0)$ as the extrapolation of its movement before its actual starting time.) Likewise, $m_k(0)$ must lie on $\overline{e_r(0)}$. By assumption, $m_k(0)$ lies left to the speed ray of m and right to the speed ray of m_1. Hence, the arms of m_k span a smaller angle (on the propagation side of the motorcycle) than the one of m. It follows that $e(0)$ lies above $\overline{e_l(0)}$.

We denote by Δ the triangle enclosed by $\overline{e_l(0)}$ and the supporting lines of the traces of m_1 and m_k. Among all vertices of G within Δ, we denote by v a vertex that has maximum orthogonal distance to $\overline{e_l(0)}$. Note that at least the endpoints of $e(0)$ are within Δ. The one endpoint that is not incident to L has the greater orthogonal distance of both. We denote by n the propagation vector of e_l and by l the parallel line of e_l through v, see Figure 33 (a). The

vector n has unit length and is orthogonal to e_l. If there are multiple vertices of G that lie on l then we may assume that v is the left-most on l (that is, the closest to the trace of m_1).

Consider l to be a sweep line moving parallel according to the speed vector n. We show that there is always a motorcycle m' that emanates from v and which is at any time in front of the sweep line l (or just coinciding). In order to see that we distinguish two cases:

- The vertex v has two or more incident edges. Since no incident edge of v lies above l we have a motorcycle m' emanating from v, see Figure 33 (b). We denote by e'' the left arm of m'. Note that $e''(0)$ is not collinear with l; otherwise there would be another vertex of G on l which is left to v.

  We denote by s the bisector of l and e'', which consists of those points that are reached by the propagating supporting line $\overline{e''(t)}$ and the sweep line l at the same time. Any point right to s is at first reached by $\overline{e''(t)}$. Since the motorcycle m' is at any time right to s we see that m' is always in front of the moving line l.
- The vertex v is a terminal vertex. There are two motorcycles emanating from v. We denote by m' the one whose speed vector has the greater inner product with n, see Figure 33 (c). We denote by e'' the arm of m' which is parallel to the incident edge of v. (W.l.o.g. we may assume that this is the left arm. The other case is symmetric.)

  As in the first case, we denote by s the bisector of the moving line l an e''. Since m' is right to s (or just overlapping) we see that m' is never behind the moving line l.

To sum up, there is always a motorcycle m' that never falls behind the sweep line l. Next, we note that l intersects R and L in their relative interiors. We conclude the proof with the following case distinction:

1. Assume that m' reaches R or L. We first see that m_1 and m_k always stay strictly behind the sweep line l: This is easy to see for m_1 since $m_1(0)$ starts behind the sweep line l and e_l propagates with the same speed as l. (Note that $m_1(t)$ and $\overline{e_l(t)}$ are coinciding.) We assume for a moment that m_k would overtake l at some time. Since m_k started behind the sweep line l, it follows that m_k reaches p before the sweep line does. However, this is a contradiction, because m_k cannot reach p before m_1 does.

   Since m_1 and m_k stay behind the sweep line l and since m' is always in front of (or just coinciding with) l, it follows that (i) m_1 crashes into m', or (ii) m_k crashes into m', or (iii) m_1 and m_k reach p but m' was at p before. In any case, the launch of m from p is avoided.
2. Assume that m' does not reach R or L. Hence, m' crashed within Δ. The motorcycle m' did not crash in a wall. (Otherwise, there would have been a vertex of G within Δ that has a greater orthogonal distance to $\overline{e_l(0)}$.) Hence, m' crashed into the trace of a motorcycle m'', which must have started below l, because no motorcycle could have come through R or L. Hence m'' is faster than m' w. r. t. direction n. That is, m'' is always in front of the sweep line l after m' crashed into m''. We consider m'' as the new m' and apply again our case analysis. Since there is only a finite number of motorcycles, we eventually end up in Case 1 and obtain a contradiction.

□

Theorem 2.26 ([HH11c])**.** *The reflex arcs of $\mathcal{S}(G)$ are covered by $\mathcal{M}(G)$.*

Proof. The proof consists of two steps. In the first step we state and prove an essential claim, which basically states that the tilted motorcycle traces of $\mathcal{M}(G)$ are above the terrain $\mathcal{T}(G)$. This proposition is applied later on in a proof by contradiction for the actual theorem.

(1) Let m be a motorcycle and $p \in \mathbb{R}^2$ a point on the trace of m. If all valleys of $\mathcal{T}(G)$ are covered by tilted motorcycle traces up to the height of $\hat{m}$ at p then the height of $\hat{m}$ is greater or equal to the height of $\mathcal{T}(G)$ at p. Equality is attained if and only if there is a valley of $\mathcal{T}(G)$ that corresponds to $\hat{m}$ and which exists until p.

We denote by e the right arm of m and we denote by $m_1, \ldots, m_k$ the motorcycles of the right-most ancestor chain of m such that $m_1(0)$ is incident to $e(0)$ and m_k equals m, see Figure 34 for $k = 2$. Furthermore, we denote by p_i the endpoint of the trace of m_i. We arrive at the following observations:

- The motorcycles $m_1, \ldots, m_k$ share the same right arm e. As a consequence, the tilted traces $\hat{m}_1, \ldots, \hat{m}_k$ lie on a plane, namely the supporting plane of the terrain face $\hat{f}(e)$ of the wavefront edge e.
- The angle between $e(0)$ and the trace of m_1 and between the traces of m_i and m_{i+1}, with $1 \leq i \leq k-1$, are at most 180° by Lemma 2.25. However, for any $1 \leq i \leq k$ the motorcycle m_i spans with its right arm e an angle of at least 90° by definition of the motorcycles.

Let us denote by $\mathcal{T}$ the polygonal chain that is defined by the intersection of $\mathcal{T}(G)$ with a vertical curtain that is put on the union of the motorcycle traces of $m_1, \ldots, m_k$. Claim (1) states that the height of $\hat{m}_k$ is greater than or equal to the height of $\mathcal{T}$ at p. In order to prove this claim it suffices to show that the slope of $\mathcal{T}$ is at any interior point of a trace of m_i at most the slope of the tilted trace $\hat{m}_i$. The following proof is an induction-type proof. We show (i) that $\mathcal{T}$ is convex within the interior of motorcycle traces, and (ii) that the slope constraint is maintained when migrating from one trace to the next.

(i) Due to the existence of m_1 there is a reflex wavefront vertex that started from $m_1(0)$. The wavefront vertex traces a reflex straight skeleton arc and has the same speed as m_1 by our definition of the motorcycles. Hence $\mathcal{T}$ and $\hat{m}_1$ start with the same slope. Let us consider the part T of $\mathcal{T}$ that lies above the trace of m_1. If there is a reflex vertex in T then we consider the one whose projection q on the plane is closest to $m_1(0)$. Obviously there would be a valley of $\mathcal{T}(G)$ at q. By assumption there would also be a motorcycle trace covering this valley at q. Since $\hat{m}_1$ is above (or just at the same height as) this trace, it follows that m_1 would have crashed at q. This is a contradiction. Hence the slope of $\mathcal{T}$ is non-increasing above the trace of m_1. The same arguments suffice to show that $\mathcal{T}$ is non-increasing above the trace of m_i if $\mathcal{T}$ was below $\hat{m}_i$ at p_{i-1}.

(ii) Next, we show that if the slope of $\mathcal{T}$ is less than or equal to the slope of $\hat{m}_i$ before p_i, then the slope of $\mathcal{T}$ is less than or equal to the slope of $\hat{m}_{i+1}$ after p_i. We denote by e' the wavefront edge which defines $\mathcal{T}$ after p_i.

The slope of $\mathcal{T}$ before p_i can be expressed by the angle between the trace of m_i and e'. That is, the slope increases monotonically as the corresponding angle increases. Likewise, we can express the slope of $\mathcal{T}$ after p_i by the angle between the trace of m_{i+1} and e'. Moreover, we can express the slopes of $\hat{m}_i$ resp. $\hat{m}_{i+1}$ by the angle between m_i and e resp. m_{i+1} and e. Hence, we can rephrase our assertion: If the angle between m_i

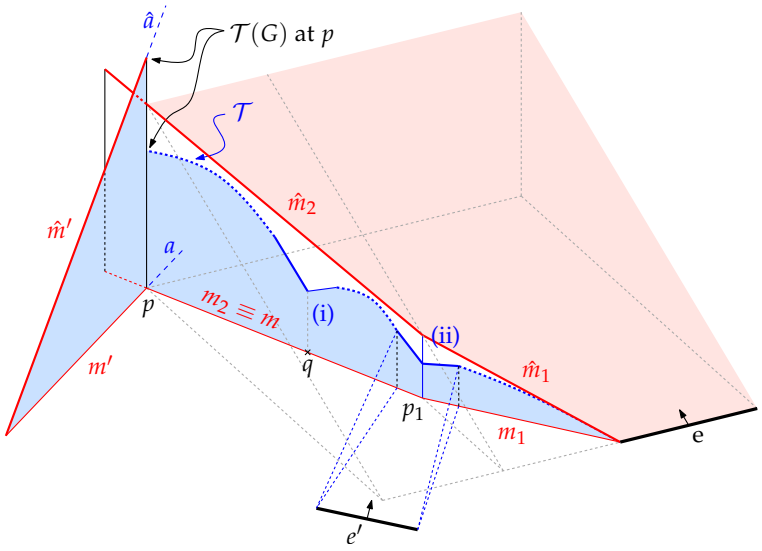


Figure 34: Proof of Theorem 2.26. The terrain $\mathcal{T}(G)$ is always below the tilted motorcycle traces $\hat{m}_k$. The proof shows that the slope of the terrain (blue) is at any point at most the slope of the tilted motorcycle traces (red) since the situations (i) and (ii) (shown in solid red and described in the proof) do not occur.

and e' is smaller than the angle between m_i and e then the angle between m_{i+1} and e' is smaller than the angle between m_{i+1} and e.

Let us consider Figure 35. We denote by l the bisector between e and e' on the left side and by r the bisector on the right side. Hence, we have to prove that m_{i+1} lies right to l and left to r. Our premise states that the angle between e' and m_i is less than or equal to the angle between e and m_i. We denote by e_l the left arm of m_i and by t_i the time when e reaches p_i. Assume that we rotate e' counter-clockwise around p_i until e' is parallel with $\overline{e(t_i)}$. Then l is falling onto $\overline{e(t_i)}$ and r is perpendicular to $\overline{e(t_i)}$. Vice versa, assume that we rotate e' clock-wise around p_i until e' is parallel with e_l. Then l is on the supporting line of m_i and r is on the bisector of m_i and $\overline{e(t_i)}$. We shaded the valid domains for l and r in Figure 35 accordingly. Since the angle between m_i and m_{i+1} is convex, m_{i+1} is right to the domain of l. Since m_{i+1} and e enclose an angle of at least 90°, m_{i+1} is left to the domain of r. Summarizing, for every position of e', which conforms to our initial assumption, m_{i+1} encloses a smaller angle with e' than with e.

Combining arguments (i) and (ii) yields an induction-type proof for Claim (1). Basically, we showed that the distance between $\mathcal{T}$ and the tilted motorcycle traces is (not necessarily strictly) monotonically increasing. If $\mathcal{T}$ and the tilted motorcycle traces are overlapping until p then equality for the height of $\hat{m}$ and $\mathcal{T}(G)$ at p is attained. If $\mathcal{T}$ leaves the tilted motorcycle traces at some point then $\mathcal{T}(G)$ is strictly below $\hat{m}$ at p.

We now return our attention to Figure 34 and use Claim (1) in a proof by contradiction of Theorem 2.26. Assume that there is a reflex arc a in $\mathcal{S}(G)$ that is only partially covered by a motorcycle trace m'. Hence, m' crashed into a motorcycle m. We denote by p the crashing

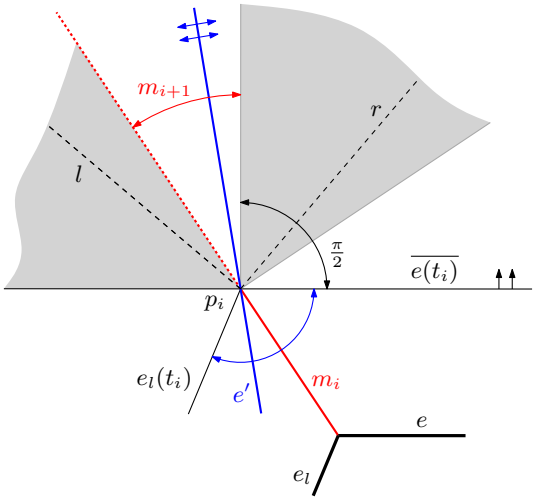


Figure 35: Proof of Theorem 2.26, case (ii). The shaded areas depict the valid domains for l and r and m_{i+1} is at any time right to l and left to r.

point of m'. Without loss of generality we assume that the height of $\hat{m}'$ at p is lowest. (By this assumption we can assume that a is at least partially covered. If a would not be covered at all, then a was not incident to G and at least one of its reflex ancestor arcs was not covered completely.) Hence, all valleys of $\mathcal{T}(G)$ are covered by motorcycle traces up to the height of $\hat{m}'$ at p. By Claim (1) we know that $\mathcal{T}(G)$ is below $\hat{m}$ at p. On the other hand, we know that $\mathcal{T}(G)$ has the same height as $\hat{m}'$ at p. (See the left side of Figure 34.) This contradiction finally concludes the proof. □

Theorem 2.26 extends Theorem 2.11 by Cheng and Vigneron [CV07]. In their proof they assumed that no two motorcycles crashed into each other. The idea of their proof is incorporated into the proof of Case (i) of Claim (1). Claim (1) is a useful tool for its own, so we can cast it to the following corollary.

Corollary 2.27. *Let m be a motorcycle of $\mathcal{M}(G)$ and $p \in \mathbb{R}^2$ a point on the trace of m. The height of $\hat{m}$ is greater or equal to the height of $\mathcal{T}(G)$ at p. Equality is attained if and only if the valley of $\mathcal{T}(G)$, which corresponds to m, exists until p.*

2.4.3 The lower envelope based on the generalized motorcycle graph

Let us revisit the discussion concerning the alternative characterization of the straight skeleton $\mathcal{S}(G)$, by using a lower-envelope representation of the terrain $\mathcal{T}(G)$ from Section 2.1. Recall that Theorem 2.7 by Eppstein and Erickson [EE99] provides a lower-envelope representation for $\mathcal{T}(G)$, but their slabs depend on the length of the reflex arcs of the straight skeleton. On the other hand, Theorem 2.14 by Cheng and Vigneron [CV07] provides a lower-envelope representation of $\mathcal{T}(P)$ that does not depend on the straight skeleton, but their representation can only be applied to simple non-degenerate polygons P with holes.

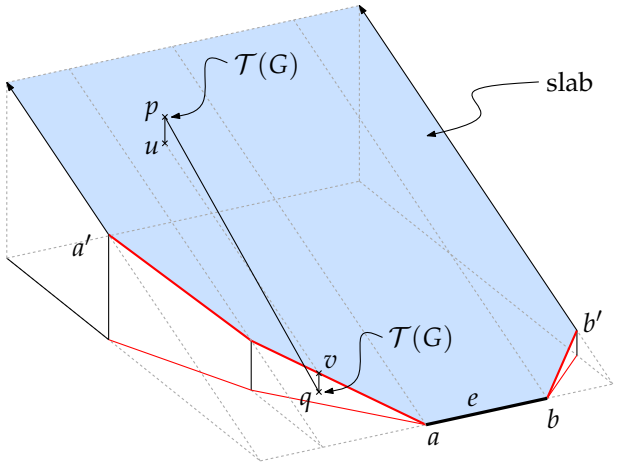


Figure 36: The shaded area illustrates the slab defined by the wavefront edge e. The slab lies on the supporting plane of $\hat{f}(e)$ and is bounded from below by the tilted motorcycle traces that have e as an arm.

Summarizing, no lower-envelope representation is known for the terrain $\mathcal{T}(G)$ of planar straight-line graphs G, which does not depend on the straight skeleton.

In the following we extend the slab construction, which is based on the motorcycle graph, to the generalized motorcycle graph, see Figure 36.

Definition 2.28 (lower envelope)**.** Let e denote a wavefront edge of a planar straight-line graph G and let a and b denote the endpoints of $e(0)$. If there is a motorcycle that starts at a and has the right arm e then we consider the whole chain of tilted motorcycles traces, starting at a and ending at a', whose right arms are e. If there is no such motorcycle then we define $a' := a$. Likewise we consider the chain of tilted motorcycle traces starting at b and ending at b' whose left arms are e. We define a slab for each e that is contained in the supporting plane of the terrain face $\hat{f}(e)$ and which is bounded from below by e and the motorcycle traces mentioned above. At the endpoints a' and b' the slab is bounded by rays which are perpendicular to $\overline{e(0)}$. We denote by $L(G)$ the *lower envelope* of the these slabs.

Theorem 2.29 ([HH11c])**.** *The lower envelope $L(G)$ is identical to $\mathcal{T}(G)$.*

Proof. Each face $\hat{f}(e)$ of $\mathcal{T}(G)$ is contained in the corresponding slab of e. It follows that no point of $L(G)$ is above $\mathcal{T}(G)$ and it remains to show that no point of $\mathcal{T}(G)$ is above $L(G)$. Assume, to the contrary, that a point $p \in \mathcal{T}(G)$ is above a slab of an edge e, see Figure 36. We project p down to the slab and denote the projection point by u. Then we project u down along the steepest descent of the slab until we hit e or one of the tilted motorcycle traces and obtain the point v. If v is on a tilted trace then Corollary 2.27 implies that we can project v down to $\mathcal{T}(G)$ and obtain a point q. If v is on $e(0)$, we set $q := v$. Since the line between u and v has slope 1, the slope of $\overline{pq}$ is greater than 1. This is a contradiction to Lemma 2.1. □

As mentioned above, Theorem 2.29 extends Theorem 2.14 by Cheng and Vigneron [CV07] to planar straight-line graphs G. However, Theorem 2.29 also extends Theorem 2.7 by Epp-

stein and Erickson [EE99] since $L(G)$ is based on $\mathcal{M}(G)$ and does not depend on the length of the reflex arcs of $\mathcal{S}(G)$. This difference, however, is essential when we attempt to compute $\mathcal{S}(G)$ via a lower-envelope computation, see below. Besides, Theorem 2.29 provides an alternative, non-procedural way to define $\mathcal{S}(G)$ as the lower envelope of partially linear functions, as discussed in Section 2.1.

COMPUTING $\mathcal{S}(G)$ USING GRAPHICS HARDWARE Theorem 2.29 admits a simple method to render $\mathcal{T}(G)$ without the knowledge of $\mathcal{S}(G)$. One first computes the motorcycle graph $\mathcal{M}(G)$ by a conventional algorithm (on the CPU) and then constructs the slabs as illustrated in Figure 36. We paint each slab with a different color. By rendering the set of slabs, while looking at them from below, one obtains an image which shows $L(G)$. Hence, by employing techniques described by Hoff et al. [HCK+99], one can compute the straight skeleton of planar straight line graphs using graphics hardware. The idea is that each face $\hat{f}(e)$ corresponds to a set of pixels of the same color.

2.5 THE GENERAL WAVEFRONT-TYPE ALGORITHM

2.5.1 Details of the general algorithm

Let us recall the basic building blocks of the simple algorithm of Section 2.3. Firstly, we require that the motorcycle graph covers the reflex arcs of the straight skeleton, which is guaranteed by Theorem 2.26. Secondly, we require that the overlay of the motorcycle graph and the input graph results in a convex tessellation of the plane.

Definition 2.30 (extended wavefront)**.** Let G denote a planar straight-line graph. We define by $\mathcal{M}(G,t)$ those parts of $\mathcal{M}(G)$ which have not been swept by $\mathcal{W}(G,t')$ for $t' < t$. The *extended wavefront* $\mathcal{W}^*(G,t)$ is defined as the overlay of $\mathcal{W}(G,t)$ and $\mathcal{M}(G,t)$ by splitting the edges of $\mathcal{W}(G,t)$ at the intersection points accordingly.

Lemma 2.31. *For any $t \geq 0$ the set $\mathbb{R}^2 \setminus \bigcup_{t' \in [0,t]} \mathcal{W}^*(G,t')$ consists of open convex faces only.*

Proof. The assertion follows directly from Lemma 2.25 and the fact that each reflex angle at a reflex wavefront vertex is split by a motorcycle trace. □

We adopt the terms *resting Steiner vertex* and *moving Steiner vertex* from Section 2.3. In addition to these terms we refer to a vertex of $\mathcal{W}^*(G,t)$ that marks the crash of two or more motorcycles as *multi Steiner vertex*, see Figure 37.

The basic algorithm from Section 2.3 remains the same. Lemma 2.31 guarantees that any topological change in the extended wavefront is indicated by the collapse of an edge of the extended wavefront to zero length. We start with the initial extended wavefront $\mathcal{W}^*(G,0)$. For each edge e of $\mathcal{W}^*(G,0)$ we insert an event into a priority queue Q if the collapsing time of e is positive. The events are prioritized by their time of occurrence. After the initialization we fetch from Q one event after the other and process it. That is, for each event we apply local modifications of the extended wavefront and maintain the priority queue Q accordingly. (Note that if we use a maximizing heap as the underlying data structure for Q we may also

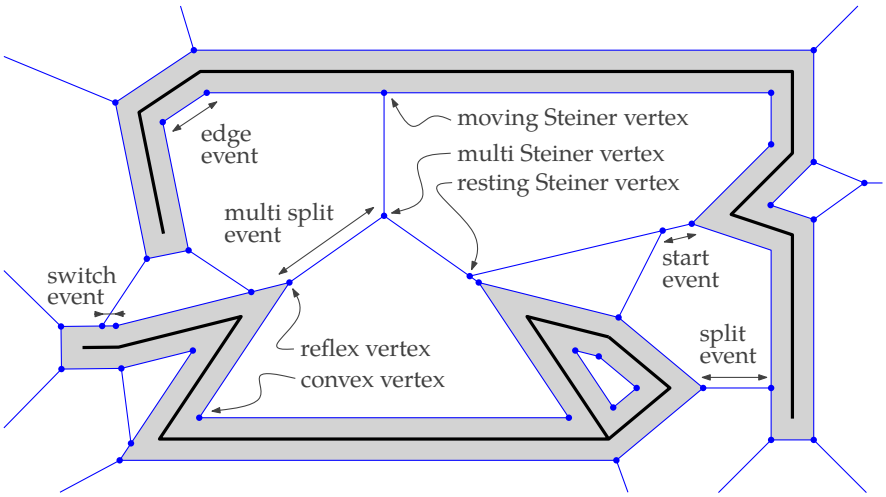


Figure 37: A planar straight-line graph (bold) and the extended wavefront (dashed) after some time. The area already swept by the wavefront is shaded.

remove any element in $O(\log n)$ time if we already have a pointer to the element.) In the following we are discussing the different types of events.

- **(Classical) edge event**: The edge e between two convex vertices u and v collapsed, see Figure 38 (a). We remove e from the graph and merge the vertices u and v to a single convex vertex w. The two edges incident to w determine the new velocity of w. Consequently, we need to recompute the collapsing times of the two incident edges of w and adapt the entries in Q.

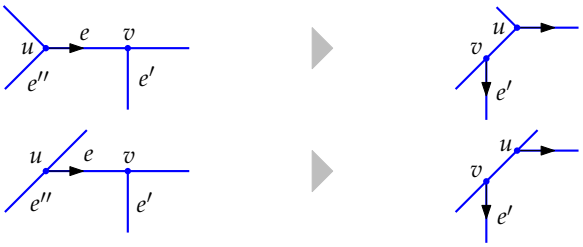
  Finally, we add the two convex straight-skeleton arcs that were traced out by u and v to the straight-skeleton graph. Altering $O(1)$ entries in Q costs us $O(\log n)$ time.
- **(Classical) split event**: The edge e between a reflex vertex u and a moving Steiner vertex v collapsed, see Figure 38 (b). Note that v is the Steiner vertex that corresponds to the crash of the motorcycle whose trace covers the reflex arc that is traced out by u.

  We denote by u_l resp. v_l the two vertices which are incident to u resp. v and left to the trajectory of u. The vertices u_r, v_r are denoted likewise. We remove the edge e and merge the vertices u_l and v_l to a new convex vertex w_l. Its velocity is determined by the two incident edges. As a special case we check whether the two incident edges of w_l are parallel. This means that the entire convex face of $\mathcal{W}^*(G,t)$, to which w_l belongs, collapsed at the time when the split event occurred. The right side, with the vertices u_r and v_r, is processed likewise.

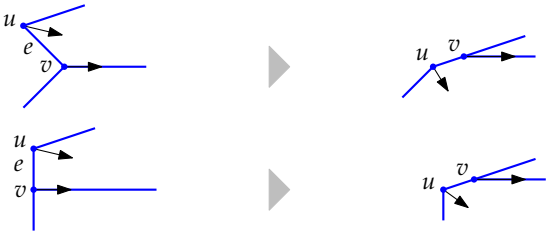
  Since u crashed at this split event, we need to add the reflex arc that is traced out by u, to our straight-skeleton structure. Altering $O(1)$ entries in Q costs us again $O(\log n)$ time.

(a) edge event

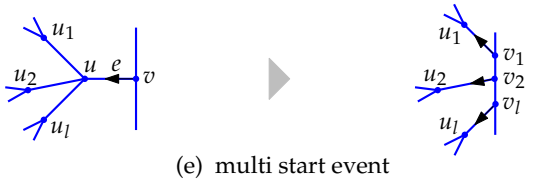
(b) split event



(c) start event



(d) switch event



(e) multi start event

(f) multi split event

Figure 38: Different types of events that occur during the propagation of the extended wavefront.

- **Start event**: The edge e that connects a resting Steiner vertex v and a reflex vertex or a moving Steiner vertex u collapsed, see Figure 38 (c). The start event occurs since the vertex u moves along the edge e, which lies on the motorcycle trace that corresponds to the vertex u. The vertex v remained on this motorcycle trace and now becomes a moving Steiner vertex by sliding along one of the two other incident edges of u, say e''. While the velocity of u remains the same, we have to assign a new velocity to v and recompute the collapsing times of the incident edges. We consider the situation where e'' and e' overlap as a special case. In this case the entire convex part collapsed to zero area.

  A start event does not immediately contribute to the straight-skeleton structure. However, altering the $O(1)$ entries in Q costs us $O(\log n)$ time.
- **Switch event**: The edge e between a convex vertex u and a reflex vertex or a moving Steiner vertex v collapsed, see Figure 38 (d). That means that the vertex u migrates from one convex face of the extended wavefront to a neighboring one, by jumping over the vertex v. As a consequence, the vertex v now splits the opposite edge incident to u. While the vertex u maintains its velocity, we need to recompute the velocity for the vertex v.

  Similar to start events, a switch event does not directly contribute to the straight skeleton. Again it requires $O(\log n)$ time to update the corresponding entries in the priority queue Q.
- **Multi start event**: The edge e that connects a moving Steiner vertex v and a multi Steiner vertex u collapsed, see Figure 38 (e). That means that the multi split event that corresponds to the multi Steiner vertex u will not happen, because the wavefront swept over the vertex u before.

  Let us denote by $u_1, \ldots, u_l$ the other vertices that are incident to u. Assume that the edges $e_1, \ldots, e_l$, which connect u with $u_1, \ldots, u_l$, respectively, appear in counter-clockwise order at u such that the edge e is between e_1 and e_l in the mentioned order. We remove the vertices u and v and introduce new moving Steiner vertices $v_1, \ldots, v_l$ such that v_i moves towards u_i, with $1 \leq i \leq l$. The new vertices $v_1, \ldots, v_l$ are aligned on the supporting line of the two other edges that are incident to v. We add new edges uv_i for $1 \leq i \leq l$ and v_iv_{i+1} for $1 \leq i \leq l-1$. Besides u, we have two additional vertices that were adjacent to v. We connect these two vertices to v_1 and v_l, respectively.

  The multi start event does not directly contribute to the straight-skeleton structure. Computing the collapsing times of the new edges and adapting the collapsing times of old edges requires $O(l \log n)$ time.
- **Multi split event**: The edges $e_1, \ldots, e_l$ that connect the reflex vertices $v_1, \ldots, v_l$ with the multi Steiner vertex u collapsed at the same time, see Figure 38 (f). We assume that $e_1, \ldots, e_l$ appear in counter-clockwise order at u. The multi Steiner vertex u exists, since two or more motorcycles simultaneously crashed at this location. Note that it is possible that a new motorcycle was emanated from this location and hence there could be an additional edge e incident to u. This edge e lies on the motorcycle trace of the emanated motorcycle, see Figure 31. If this is the case we assume that e is between e_1 and e_l. Also note that in this case the angle from e_l to e_1 is reflex.

Each consecutive pair $u_i u_{i+1}$, with $1 \leq i \leq l-1$, gives rise to a new convex vertex similar to an ordinary split event. This new convex vertex is incident to two edges that were formerly the right arm of u_i and the left arm of u_{i+1}, respectively. Similar to the ordinary split event, we check whether the two incident edges are overlapping. In this case, the whole convex face of the extended wavefront collapsed as the multi split event happened.

If the angle from e_l to e_1 is also convex then there is no additional edge e incident to u and we proceed as above. Otherwise we have an additional edge e incident to u. If the right arm of u_l and the left arm of e_1 span a convex angle than the edge e and the old edge e_l are collinear, cf. Figure 31 (d). We remove the edge e and proceed as above. Otherwise, the right arm of u_l and the left arm of e_1 span a reflex angle. Then the edge e lies on their bisector and we introduce a new reflex vertex that drives on the edge e, cf. Figure 31 (c).

Each u_i, with $1 \leq i \leq l$, traced out a reflex straight-skeleton arc, which we add to our straight-skeleton structure. Computing the collapsing times of the new edges and altering the corresponding entries in Q takes $O(l \log n)$ time.

After the last event occurred, the extended wavefront has the shape of a polygon circumscribing the graph G. From each vertex v of this polygon that has a reflex angle on the outer side, there is an incident edge which reaches to infinity and which corresponds to a motorcycle that escaped. We add these infinite edges as reflex arcs to our straight skeleton structure and connect the infinite end nodes in a circular manner. This allows us to assume that the boundary of any straight-skeleton face $f(e)$, including the unbounded faces, connects one endpoint of e with the other endpoint of e. We also refer to the discussion on unbounded faces and infinite arcs in Section 1.2.2.

2.5.2 Runtime analysis

Let us assume that the motorcycle graph $\mathcal{M}(G)$ of G is already given. In order to create the extended wavefront $\mathcal{W}^*(G,0)$ at time zero we first create $\mathcal{W}(G,0)$: We start with an edge e and a vertex v incident to e. We walk around the corresponding connected component by setting e to the next counter-clockwise edge of e at v and by setting v to the opposite vertex of the updated e. We continue this procedure until we again arrive at our starting edge. At each step, we duplicate the current edge e and consider its duplication as the wavefront edge at one side of e. At any time when we wrap around at a terminal vertex, we add an additional wavefront edge. After we arrive again at the starting edge, we repeat the whole procedure for any edge of G that has not yet a duplicate on either side. This allows us to create $\mathcal{W}(G,0)$ in $O(n \log n)$ time for an n-vertex graph G. Next, we insert $\mathcal{M}(G)$ into $\mathcal{W}(G,0)$ in order to obtain $\mathcal{W}^*(G,0)$. Hence, every motorcycle that crashed into a wall — that is, into an edge e of G — splits the corresponding wavefront duplicate of the input edge e. We need to take into account that multiple motorcycles may crash into the same side of an input edge e. We sort the motorcycles along e and split the corresponding wavefront edge accordingly. To sum up, we can construct $\mathcal{W}^*(G,0)$ in $O(n \log n)$ time if $\mathcal{M}(G)$ is known.

The number of edge events is in $O(n)$ and the number of split and start events is in $O(r) \subset O(n)$, where r denotes the number of reflex wavefront vertices in the initial wave-

front $\mathcal{W}(G,0)$. However, as discussed in Section 2.3, the number k of switch events is in $O(nr) \subset O(n^2)$. Each multi start event and multi split event is handled in $O(l \log n)$ time, where the number l corresponds to the number of motorcycles that gave rise to this event. The sum of all values l among the multi start and multi split events is therefore in $O(r) \subset O(n)$.

Lemma 2.32. *Let G denote a planar straight-line graph G with n vertices, where r denotes the number of reflex wavefront vertices in the initial wavefront $\mathcal{W}(G,0)$. If $\mathcal{M}(G)$ is known then our algorithms runs in $O((n+k)\log n)$ time and $O(n)$ space, where $k \in O(nr) \subset O(n^2)$ denotes the number of switch events occurred.*

As already mentioned in Section 2.3.3, the $O(n^2)$ bound for the number of switch events is tight. However, we want to recall that it appears to be very unlikely that more than $\Omega(n)$ switch events actually occur for practical applications and hence an $O(n \log n)$ runtime can be expected in real world. In Section 2.5.4 we present extensive experimental results on 13 500 datasets of different characteristics that demonstrate an $O(n \log n)$ runtime on virtually all of our datasets.

In order to compute the motorcycle graph $\mathcal{M}(G)$, we need an algorithm that supports the dynamic insertion of motorcycles during the simulation. Hence, we cannot apply the algorithm by Cheng and Vigneron, cf. Section 1.4.2.4, which needs to compute the $1/\sqrt{r}$-cutting a-priori. However, the algorithm by Eppstein and Erickson [EE99], cf. Section 1.4.2.3, employs dynamic data structures which are capable of adding new motorcycles during the simulation. Hence, in theory, we can compute the motorcycle graph $\mathcal{M}(G)$ in $O(r^{17/11+\epsilon})$ time and space. In Section 3.3 we will discuss a practical algorithm that runs in $O(r \log r)$ time on average, under the assumption that the motorcycles are distributed uniformly enough.

2.5.3 Details of the implementation Bone

We casted our algorithm into the implementation Bone. Bone is written in C++ and uses ordinary double-precision floating-point arithmetic according to IEEE-754. For standard data structures, such as stacks, queues, maps (red-black trees), etc., we use the Standard Template Library. The motorcycle graph $\mathcal{M}(G)$ is computed by our motorcycle graph code Moca, see Section 3.3. One central component of Bone is a kinetic straight-line graph that assigns to each vertex a speed ray (the position at time zero and a velocity). Furthermore, each non-isolated vertex has an index to an incident edge and each edge has, for both endpoints, references to the next clockwise and counter-clockwise edges. The kinetic graph models the extended wavefront and does not maintain planarity. It is in the responsibility of the event handling code to maintain the proper topology of the kinetic graph. However, this graph structure allows us to easily remove edges and reconnect vertices, which is frequently done in the event handling procedures.

For the actual implementation it turned out that is advantageous to introduce a new type of vertex, the so-called *multi convex vertex*. In a degenerate case, it could happen that a convex wavefront vertex and a moving Steiner vertex move along the same trajectory. For instance, consider a symmetric non-convex 4-gon as input, where the motorcycle within the 4-gon crashes exactly into the opposite convex vertex. In this case it is important, for practical reasons, to merge the convex vertex and the Steiner vertex into a multi convex vertex.

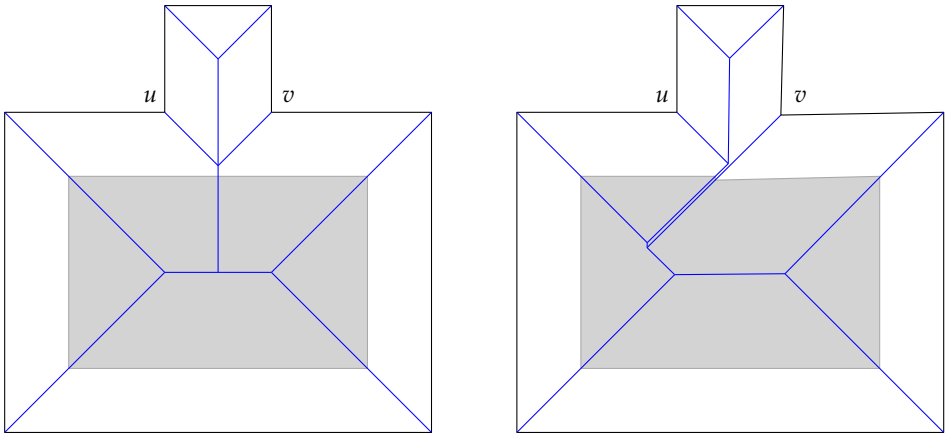


Figure 39: An arbitrarily small perturbation of a vertex v significantly changes the straight skeleton. Left: v is involved in a vertex event with the vertex u. Right: v has been slightly dislocated to the interior of the left polygon. Since v is slightly faster than u, the vertex event is avoided and the reflex arc incident to v is dramatically longer. The corresponding offset polygons are depicted in gray. Note that the offset polygon in the right subfigure contains a reflex vertex at the top.

For real-world implementations of geometric algorithms, it is advisable to implement some a posteriori checks that test whether the result fulfills some basic but necessary properties in order to be correct. Bone follows that strategy and provides a posteriori checks that test whether the boundary of the faces are connected.

DISCONTINUITY AND ϵ-BASED COMPARISONS We already mentioned the unpleasant property of the straight skeleton $\mathcal{S}(G)$ to be very sensitive for changes on the input G multiple times. (For instance, this fact made it necessary in introduce the non-degeneracy assumption by Cheng and Vigneron, cf. Section 1.2.4.) Eppstein and Erickson [EE99] presented a simple example, where a vertex event occurs and for which an arbitrary small perturbation at certain input vertices leads to a significantly different straight skeleton. We illustrate a very similar example in Figure 39. Note that dislocating the vertex v in the right subfigure introduces a proper reflex vertex in the gray offset polygon, while in the left subfigure the offset polygon has a rectangular shape with the top edge split into two collinear edges.

The fact that the straight skeleton $\mathcal{S}(G)$ does not continuously depend on the input graph G — say, in terms of the Hausdorff distance — has important practical implications. First of all, we cannot apply perturbation techniques in order to resolve special cases in various algorithms, as already mentioned by Eppstein and Erickson [EE99]. Secondly, we have to be cautious when we consider the limit of a sequence of straight skeletons. For instance, image we define the wavefront emanated by an isolated vertex by approximating the vertex by small squares or small segments. (Recall the discussion on the wavefront emanated from a isolated vertex in Section 1.2.2.) The limit of the straight skeletons does not necessarily result in the desired structure and may depend on the actual sequence of input graphs. Thirdly,

the question arises how to determine whether two motorcycles or two reflex vertices meet each other at exactly the same time.

- If we use ordinary double-precision floating-point arithmetic then we need to apply ϵ-based comparisons. That is, we define a small positive constant ϵ and in order to test whether the numbers a and b are equal we tests whether $|a - b| \leq \epsilon$.
- In order to avoid well-known problems introduced by ϵ-based comparisons, one could apply so-called exact predicates. The decision whether two values a and b are equal is based on the expression trees for the values a and b and the concept of so-called root-bounds [Yap04]. That is, one can precisely decide whether $|a - b|$ is zero, by evaluating the term $|a - b|$ with finite but arbitrary precision. The root bounds tell how precisely the term needs to be evaluated.

Let us consider the application of roof construction, cf. Section 1.2.3. Assume that we implement a roof-construction software based on straight skeletons, which reads the wall footprint of a house from an input file. Since the straight skeleton is sensitive to small errors on the input we would require that the input file is capable of carrying exact coordinates. This basically means that we need to save expression trees of all coordinates in the input files. However, this seems to be infeasible for real-world applications:

- The file size would skyrocket and, moreover, the expression trees tend to grow in size as the input gets more and more complex, because the coordinates of new vertices tend to depend on the positions of previous vertices.
- Standard file formats, like the DXF format, do not support the embedding of expression trees for the coordinates in a standardized way.
- We would require that common CAD software uses exact arithmetic kernels for their own calculations (e. g., snapping to intersection points). In the end, we would require that the entire work-flow uses exact arithmetic.

However, if we assume that the coordinates in the input files are imprecise due to the limited precision or ordinary floating-point numbers, we are basically forced to resort to ϵ-based comparisons for real-world applications. Otherwise, it is virtually impossible to construct non-trivial input data, for which we intend that a vertex event happens. Assume we would like to construct a symmetric wall footprint as in Figure 39 (a), which would produce a roof, where two valleys meet in a common endpoint. However, due to numerical imprecisions, we instead obtain a roof that is based on Figure 39 (b).

Note that for Voronoi diagrams the problem of imprecise input is not as severe as for straight skeletons, because the error introduced to the Voronoi diagrams is, roughly speaking, of the same magnitude as the error embedded in the input values.

2.5.4 Experimental results and runtime statistics

In this subsection we present experimental results for our implementation Bone and the straight-skeleton implementation that is shipped with CGAL 3.8, see Section 1.4.3. As already mentioned, Bone is able to process planar straight-line graphs as input, but the implementation in CGAL can only process polygons with holes. This is the reason why we restrict the following runtime statistics to datasets that contain polygons with holes only.

Since BONE computes the straight skeleton inside and outside the input polygon we call the straight-skeleton routines of CGAL outside the polygon, inside the polygon and also inside the holes, if any exist.[2]

We ran our runtime tests on a about 13 500 datasets, comprising realistic and contrived data of different characteristics, including mechanical designs, printed circuit boards, font outlines, random polygons generated by RPG [AH96], space filling curves, and fractal datasets. The size of the datasets ranges from only a few dozen vertices up to several millions.

Our experiments were carried out on a Linux machine with a 64-bit kernel, running on an Intel Core i7 processor clocked at 3.33 GHz. Note that this processor provides several cores, but neither BONE nor the straight-skeleton implementation from CGAL is multi-threaded. We avoided a falsified system behavior — e. g., caused by the invocation of the out-of-memory killer of Linux, freeing of all file system buffers, etc. — by restricting the memory utilization of BONE resp. the CGAL code to 6 GiB, using the `ulimit` command. Furthermore, we restricted the runtime for a single dataset to 10 minutes. The time consumption in order to compute the straight skeleton was measured by considering the user-space time consumption retrieved by the C-function `getrusage`.

Figure 40 shows four plots with identical axes and in each plot a blue dot represents the runtime of a single dataset. Note that both axes use a logarithmic scale. The x-axis shows the size n of the dataset and the y-axis illustrates the runtime for a dataset in seconds. For illustrative reasons we divided the runtime for each dataset of size n by the factor $n \log n$. Hence, a horizontal line in the plot corresponds to a runtime complexity of $n \log n$ and a line with slope 1 corresponds to a runtime complexity of $n^2 \log n$.

In subfigure (a) we plotted the runtime statistics for BONE. As predicted, BONE exhibits an $n \log n$ runtime behavior on the vast majority of our datasets. All dots in the gray area correspond to an actual runtime of 10 to $30 \cdot n \log n$ µs, where n varies from 60 up to $2 \cdot 10^6$. The motorcycle graph $\mathcal{M}(G)$ is computed by MOCA, see Section 3.3. This implementation exhibits an $O(n \log n)$ runtime for well distributed input. We extensively investigate the runtime of MOCA in Section 3.3.3. Our theoretical runtime analysis in Section 2.5.2 at first focuses on the assumption that the motorcycle graph is already available. For this reason, we plotted the runtime consumption of BONE, without the time consumed for the computation of the motorcycle graph $\mathcal{M}(G)$, in subfigure (b). We observe that the few outliers in subfigure (a) were virtually all caused by the computation of the motorcycle graph. Hence, an actual runtime of $O(n \log n)$ holds for basically all datasets from our database.

In subfigure (c) we plotted the runtime of CGAL using an arithmetic kernel with exact predicates but inexact constructions. By default, the CORE [Cor] library is employed as the arithmetic backend. Using exact predicates should guarantee that the topology of the straight skeleton is correct, even if the locations of the nodes are not necessarily exact.[3] However, using exact constructions makes the runtime consumption to skyrocket. That is, datasets with a few hundred vertices lead to runtimes in the range of minutes. We see that the runtime consumption of CGAL is mostly in the gray area that represents runtimes between 0.17 to $1.7 \cdot n^2 \log n$ µs. In particular, CGAL exhibits at least a quadratic runtime consumption in practice. We also observe that the runtimes of different datasets of the

2 CGAL requires to specify a maximum offset distance for the straight skeleton outside the polygon. After we scaled the input to the unit square, we set this offset distance to 100.

3 We learned this in a personal communication with F. Cacciola.

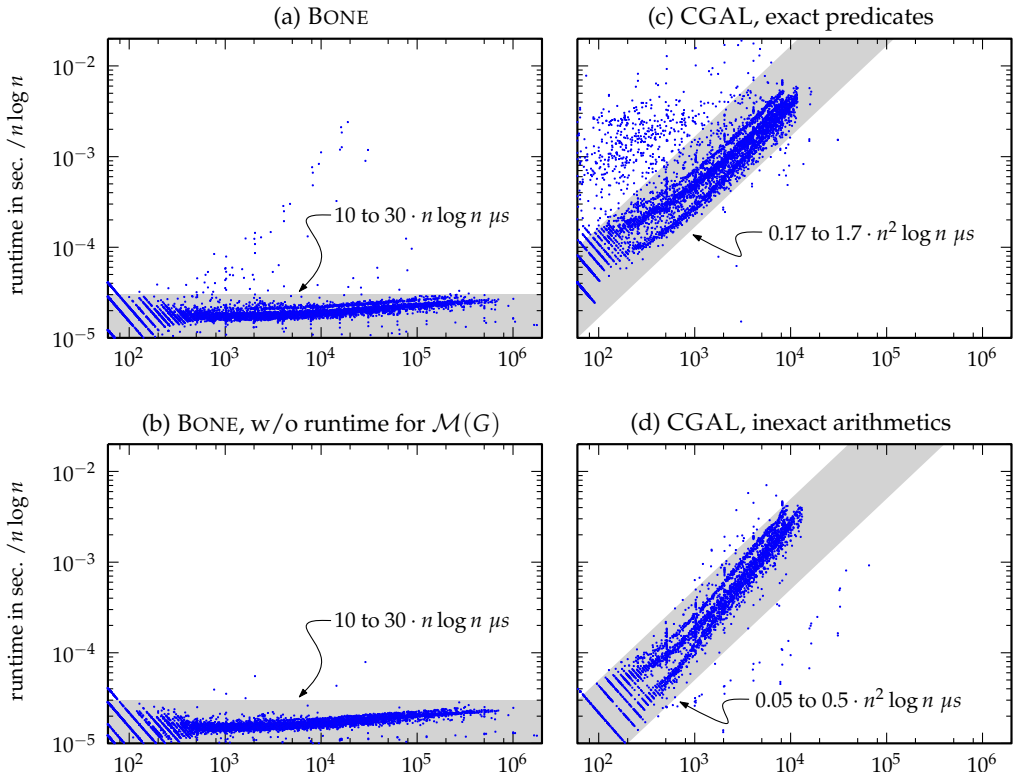


Figure 40: Every point shows the runtime for a single dataset. The x-axis denotes the number n of vertices of the input, ranging from 60 to $2 \cdot 10^6$. For illustrative reasons we divided the actual runtime in seconds by $n \log n$. Hence, a horizontal line corresponds to an $n \log n$ complexity. Points in the shaded areas correspond to runtimes as labeled.

same size vary within two decades. While many datasets with about a hundred vertices need roughly 20 ms, many other datasets lead to runtimes of roughly 20 seconds.

In subfigure (d) we plotted the runtime of CGAL using inexact arithmetics in order to (i) gain a better insight on the runtime penalty due to the exact predicates kernel and (ii) to obtain a better comparison to BONE. We first note that CGAL, with inexact arithmetics, still exhibits at least a quadratic runtime complexity. The vast majority of datasets is processed in 0.05 to $0.5 \cdot n^2 \log n$ µs. However, the big deviations for the runtimes among datasets of the same size vanished. We also observe that using inexact arithmetics increases the performance by a factor of about 3 to 100. However, we want to note that the straight-skeleton implementation of CGAL is not supposed to be used with an inexact predicates kernel.[4]

MEMORY USAGE BONE was able to handle input containing more than 10^6 vertices. However, CGAL could only handle datasets with up to about 10 000 vertices within the time and space constraints mentioned above. Interestingly, also CGAL with inexact arithmetics

4 We refer to a personal mail correspondence with F. Cacciola.

Size n	BONE		CGAL	
	MB	factor	MB	factor
256	1.44		3.77	
512	2.65	1.8x	13.4	3.5x
1 024	5.06	1.9x	51.1	3.8x
2 048	9.86	1.9x	201	3.9x
4 096	19.5	2.0x	792	3.9x
8 192	38.7	2.0x	3 197	4.0x
16 384	77.1	2.0x	12 600	3.9x

Table 3: The memory usage of BONE and CGAL on random datasets generated by RPG. Each row contains the peak heap size in megabytes and the factor by which the size changed w. r. t. the previous row.

was not able to compute datasets containing significantly more than about 10 000 vertices within the 6 GiB memory limit. We listed the memory footprint of BONE and CGAL on random datasets that were generated by RPG in Table 3. The memory usage was measured by the library LIBMEMUSAGE.SO, which is shipped with GLIBC [GLI]. This library hooks into the `malloc` calls of a process and accounts the memory usage. In Table 3, we print the peak heap size in megabytes and the relative change to the previous row. Note that the actual memory footprint of a process is larger due to ordinary memory fragmentation in the heap and, of course, the program code, libraries and the stack require a few megabytes as well. For instance, the CGAL process required in total almost 20 GiB virtual memory in order to process the last dataset with 16 384 vertices.

We observe that BONE doubles its memory footprint if the input size of the dataset is doubled. In contrast, CGAL quadruples its memory footprint if the input size is doubled, which suggests a quadratic space complexity. As we learned in a personal mail correspondence with F. Cacciola, the author of the straight-skeleton code in CGAL, the algorithm computes potential split events for all pairs of reflex wavefront vertices and wavefront edges and inserts them into a priority queue. This explains the $O(n^2)$ memory footprint and the $O(n^2 \log n)$ runtime.

We tested the reliability of BONE on additional datasets, which do not necessarily form polygons with holes. BONE is also able to sample circular arcs from input files by straight-line segments. This allows us to test BONE on basically the same datasets as we did for VRONI [HH09a]. Furthermore, BONE is able to compute offset-curves based on the straight-skeleton, which turned out to be a versatile development tool in order to find errors in the straight skeleton by visual inspections. Furthermore, BONE can export the terrain $\mathcal{T}(G)$ into a standard file format that can be read by 3D modeling software, like Blender [Ble]. A wrong straight skeleton or significant numerical errors cause visible artifacts like non-planar terrain faces or very odd-looking terrains. BONE also contains many sanity checks for the validity of the wavefront during its propagation simulation and BONE is able to perform simple a posteriori checks on the resulting straight skeleton as mentioned in Section 2.5.3. It turns out that BONE performs well on our datasets in general. However, in order to boost BONE to industrial-strength, we need to implement some fine-tuning in the presence of wavefront-

parts that simultaneously collapse in a self-parallel manner. However, the concept of the extended wavefront allows us to handle this issue on each convex faces of the extended wavefront separately, which appears to be a significant advantage.

2.6 SUMMARY

At the beginning of this chapter, the most promising approach towards a straight-skeleton implementation, in terms of implementability and real-world performance, appeared to be the triangulation-based algorithm by Aichholzer and Aurenhammer [AA98]. However, from a theoretical point of view, the best known worst-case time complexity is $O(n^3 \log n)$ for an n-vertex planar straight-line graph. In Section 2.2, we used this circumstance as an entry point to our investigations. After presenting new aspects regarding the number of flip events, we were able to show that Steiner triangulations can be used in order to completely avoid flip events. This insight motivated a new approach to a straight-skeleton algorithm for non-degenerate polygons which is based on motorcycle graphs in Section 2.3. However, in order to generalize this algorithm to arbitrary planar straight-line graphs we had to carefully extend the motorcycle graph such that essential properties, which are required for our algorithm, are preserved, see Section 2.4. This generalization of the motorcycle graph allowed us to extend the approach from Section 2.3 to arbitrary planar straight-line graphs in Section 2.5.

The resulting implementation Bone is able to handle planar straight-line graphs as input and exhibits a runtime of $O(n \log n)$ in practice. Compared to the current state-of-the-art straight-skeleton code, which is shipped with CGAL, this constitutes an improvement of a linear factor in time and space resp. a speed-up of one to two orders of magnitude for medium-sized datasets. Furthermore, CGAL is, in contrast to Bone, only able to handle polygons with holes as input. Our experimental results in Section 2.5.4 also revealed that Bone was able to process datasets containing more than a million vertices, while the implementation in CGAL was not able to handle datasets with significantly more than 10 000 vertices within the limits of 6 GiB memory.

The theoretical worst-case runtime complexity of Bone is $O(n^2 \log n)$, which is an improvement of a linear factor compared to the algorithm of Aichholzer and Aurenhammer [AA98]. However, the circumstances for which the wort-case occurs are easy to investigate for the algorithm underneath Bone, see Section 2.5. In fact, it appears to be very unlikely that the worst case actually happens for practical input, which has been confirmed by our experimental results in Section 2.5.4. Bone computed the straight skeleton for virtually all of our 13 500 datasets in 10 to $30 \cdot n \log n$ µs for datasets containing n vertices. This makes Bone the first straight-skeleton implementation that has been shown to run in $O(n \log n)$ in practice and which accepts planar straight-line graphs as input.

3 MOTORCYCLE GRAPHS

Motorcycle graphs are strongly related to straight skeletons for several reasons. First of all, the motorcycle graph extracts the essential sub problem of computing straight skeletons. Secondly, the motorcycle graph and the straight skeleton possess a strong geometric relation, which is expressed by Theorem 2.11 and Theorem 2.26. Thirdly, motorcycle graphs help us to devise algorithms and implementations for the computation of straight skeletons. At the moment, the fastest algorithm for the straight skeleton of non-degenerate polygons due to Cheng and Vigneron [CV07], see Section 1.4.2.4, and the fastest implementation, Bone, both employ the motorcycle graph. Hence, the investigation of motorcycle graphs is vital in order to obtain fast straight-skeleton algorithms and implementations and it is important to explore the motorcycle graph in order to get a better understanding of straight skeletons.

In this chapter, we develop an implementation for the computation of the generalized motorcycle graph that works fast in practice. We start with a stochastic consideration of the trace lengths in a motorcycle graph in Section 3.2. It turns out that if n motorcycles are distributed well in the unit square then the average trace length is about $O(1/\sqrt{n})$. This insight motivates a motorcycle graph implementation that is based on geometric hashing in Section 3.3. Runtime experiments show that our implementation Moca exhibits an $O(n \log n)$ runtime for most of the 22 000 datasets tested. In Section 3.4, we further investigate the geometric relation between the motorcycle graph and the straight skeleton. The results obtained in this section finally lead to a proof for the P-completeness of the straight skeleton of planar straight-line graphs and polygons with holes that is based on the P-completeness of motorcycle graphs due to Eppstein and Erickson [EE99]. The P-completeness of straight skeletons has important practical implications concerning the application of parallel algorithms. The investigations done in Section 3.2 and Section 3.3 are published in [HH11b]. A preliminary and short version was presented in [HH09b]. The results in Section 3.4 have been submitted for publication [HH11a].

- [HH09b] S. Huber and M. Held. A Practice-Minded Approach to Computing Motorcycle Graphs. In *Proc. 25th Europ. Workshop Comput. Geom.*, pages 305–308, Brussels, Belgium, Mar 2009
- [HH11b] S. Huber and M. Held. Motorcycle Graphs: Stochastic Properties Motivate an Efficient Yet Simple Implementation. *J. Exp. Algorithmics*, 2011. (in press)
- [HH11a] S. Huber and M. Held. Approximating a Motorcycle Graph by a Straight Skeleton. 2011. (submitted for publication)

3.1 PRIOR AND RELATED WORK

3.1.1 Applications of motorcycle graphs and related problems

The most prominent application of motorcycle graphs are straight skeletons. However, motorcycle graphs are also related to other geometric problems. Obviously, there exists a strong connection to ray-shooting problems. For instance, Ishaque et al. [IST09] presented an $O(n \log^2 n + m \log m)$ algorithm for m repetitive ray shooting-and-insertion operations in the plane among a set of polygonal obstacles of total size n. The generalized motorcycle graph problem, for which motorcycles need not all start at the same time, solves this problem in the following way: The polygonal obstacles are replaced by according walls and each ray is replaced by a motorcycle. By choosing the start time of the motorcycles such that no two motorcycles drive at the same time, one can simulate repetitive ray-shootings.

Eppstein and Erickson [EE99] mentioned that the art gallery algorithm by Czyzowicz et al. [CRU89] uses a geometric structure that is related to the motorcycle graph. They consider a set of straight-line segments, where each segment is growing in length and an endpoint of a segment stops propagating when it reaches another segment. The resulting structure can be interpreted as a motorcycle graph: The initial segments are considered as walls and from each endpoint is a motorcycle launched.

Eppstein et al. [EGKT08] consider motorcycle graphs on quadrilateral meshes, which model three-dimensional bodies. The idea is that motorcycles are driving on the edges of the mesh in a discrete manner and a motorcycle stops when it reaches the trace of another motorcycle. It is not exactly the same motorcycle graph problem as introduced by [EE99], but very similar in nature.

3.1.2 Prior work

At the moment, the most efficient motorcycle graph algorithm is due to Cheng and Vigneron [CV07], see Section 1.4.2.4. It computes the motorcycle graph of n motorcycles in $O(n\sqrt{n} \log n)$ time. Let us recall the main idea of their algorithm: Among the $O(n^2)$ intersections of the tracks of the motorcycles only $O(n)$ of them correspond to an actual crash event. Hence, the goal is to geometrically separate those motorcycles that do not interact. Cheng and Vigneron achieved this goal via the employment of $1/\sqrt{n}$-cuttings on the supporting lines of the traces. However, in order to reach the $O(n\sqrt{n} \log n)$ runtime complexity, the algorithm inherently relies on the fact that all motorcycles are known a priori such that the $1/\sqrt{n}$-cutting can be constructed prior to the simulation of the motorcycles movement. As a consequence, this algorithm is not suitable for the computation of the generalized motorcycle graph presented in Section 2.4.

In order to solve the generalized motorcycle graph problem, where motorcycles are allowed to emerge after the simulation started, we are in need of an algorithm that allows the dynamic insertion of motorcycles. Under this requirement, the fastest algorithm is due to Eppstein and Erickson [EE99] and runs in $O(n^{17/11+\epsilon})$ time and space, see Section 1.4.2.3. However, we want to emphasize that this algorithm is not suitable for an actual implementation due to its algorithmic complexity.

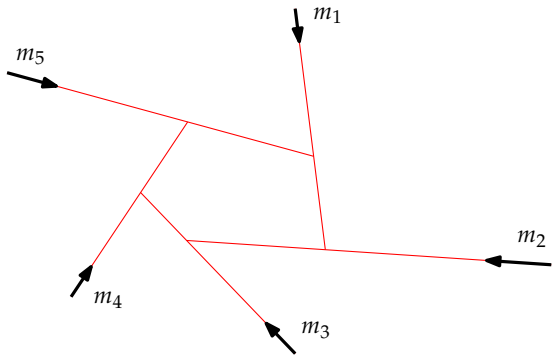


Figure 41: All motorcycles crash into each other.

A straight-forward approach for the actual computation of the motorcycle graph of n motorcycles is to simply simulate their movement in a brute-force manner. We can compute one crash event after the other by pair-wise checks among the motorcycles. By employing a priority queue Q for pending crash events, one obtains an implementation that runs in $O(n^2 \log n)$ time instead of $O(n^3)$. We refer to this algorithm as the standard priority-queue based algorithm.

To sum up, two sub-quadratic algorithms for the computation of motorcycle graphs are known, but only the algorithm by Eppstein and Erickson [EE99] can be used to compute the generalized motorcycle graph problem. Furthermore, no algorithm is known that is suitable for implementation, on one hand, and can be expected to exhibit a significantly sub-quadratic runtime for real-world applications on the other hand.

3.1.3 Geometric properties of the motorcycle graph

We interpret $\mathcal{M}(m_1, \ldots, m_n)$ as a graph and we add the infinite endpoints of the traces of escaped motorcycles to the graph as well. It follows that $\mathcal{M}(m_1, \ldots, m_n)$ contains exactly $2n$ vertices: Each motorcycle contributes a start point and an end point. (If multiple vertices coincide geometrically, we still count them as different vertices.) The number of finite vertices ranges from n to at most $2n$. The lower bound is attained if all motorcycles escape. The upper bound is attained if all motorcycles crash into each other, as illustrated in Figure 41.

Lemma 3.1. *$\mathcal{M}(m_1, \ldots, m_n)$ contains between n and $2n$ edges. The lower bound is attained if all motorcycles escape. The upper bound is attained if all motorcycles crash.*

Proof. The edge set of $\mathcal{M}(m_1, \ldots, m_n)$ results from n motorcycle traces, which are possibly split. Hence, the number of edges is at least n. In order to show the upper bound of $2n$, we denote by n_c the number of crashed motorcycles and by e the number of edges. $\mathcal{M}(m_1, \ldots, m_n)$ contains $n + (n - n_c)$ vertices of degree 1 and n_c vertices of degree 3. (Note that we obtain $n - n^c$ infinite vertices.) We charge each edge by both of its incident vertices. Considering the total number of charges results in $2e = n + (n - n_c) + 3n_c = 2n + 2n_c \leq 4n$ and hence $e \leq 2n$. The bound is tight if $n = n_c$. □

The following lemma is given by Eppstein and Erickson [EE99] without a proof. In the following, a pseudo-forest is a graph whose components are pseudo-trees and a pseudo-tree is a connected graph that contains at most one cycle.

Lemma 3.2 ([EE99]). *$\mathcal{M}(m_1, \ldots, m_n)$ is a pseudo-forest.*

Proof. We denote the number of edges of $\mathcal{M}(m_1, \ldots, m_n)$ by e. We may assume that the motorcycle graph $\mathcal{M}(m_1, \ldots, m_n)$ is connected; the general case is shown analogously. If $\mathcal{M}(m_1, \ldots, m_n)$ is not a pseudo-tree we can remove at least two edges without hurting connectedness. Since a connected graph with $2n$ vertices contains at least $2n - 1$ edges, it follows that $e - 2 \geq 2n - 1$. This is a contradiction to Lemma 3.1. □

3.2 STOCHASTIC CONSIDERATIONS OF THE MOTORCYCLE GRAPH

3.2.1 Number of intersections of bounded rays

In order to devise a motorcycle graph algorithm that runs fast in practice, it appears interesting to investigate the average trace length within the motorcycle graph $\mathcal{M}(m_1, \ldots, m_n)$ of n motorcycles. Let us recall that the tracks of the motorcycles lead to $O(n^2)$ intersections, whereas the traces produce at most $O(n)$ intersections, namely at precisely those points where a motorcycle crashed into the trace of another.

Let us assume for a moment that the motorcycles drive at unit speed, the start points $v_1, \ldots, v_n$ are distributed uniformly in the unit square $[0, 1]^2$ and the $\varphi_1, \ldots, \varphi_n$ are distributed uniformly on $[0, 2\pi)$. Directly computing the expectation of the trace length of a single motorcycle trace appears to be complicated due to the stochastic dependencies among the motorcycle traces. However, it seems to be evident that the mean trace length cannot get too large, because two traces are not allowed to intersect in the interior of each other.

Let us consider a regular $\sqrt{n} \times \sqrt{n}$ grid on the unit square. A single motorcycle m_i starts at one grid cell and, while it moves, crosses a specific number of other cells. Each cell contains on average one start point of a motorcycle. The probability that m_i crosses k cells is falling at least exponentially with k if we would flip a coin to decide whether m_i passes the motorcycle that started in each cell entered by m_i. This thought experiment would suggest that a motorcycle does not pass more than $O(1)$ grid cells and, as a consequence, that the average trace length is in $O(1/\sqrt{n})$. However, a simple coin flip does not take into account the stochastic dependencies among the motorcycles.

In order to simplify the original question, we reformulate the problem. Instead of asking for the mean trace length of the motorcycles, we ask for the number of intersection points of bounded rays, where the length of the rays is chosen at random according to the probability density function f. The idea is that if we gain some information on the number of intersections than we also gain information on the mean length of the bounded rays as well.

Theorem 3.3 ([HH11b]). *Let $v_1, \ldots, v_n$ denote n points that are uniformly i.i.d.[1] on the unit square $[0, 1]^2$. Further, by $\varphi_1, \ldots, \varphi_n$ we denote n angles i.i.d. on $D := \{\delta_1, \ldots, \delta_d\}$, with $d \in \mathbb{N}$, such that*

1 A common short-hand for "independent and identically distributed".

$\delta_i \in [0, 2\pi)$ occurs with probability p_i, where $\sum_i p_i = 1$. Next, let $L_1, \ldots, L_n$ be i.i.d. on $[0, 0.2]$ according to a probability density function f.

For each $i \in \{1, 2, \ldots, n\}$ consider a bounded ray $T_i \subset \mathbb{R}^2$ which starts at v_i, has direction angle φ_i and length L_i. We denote by $I = \sum_{i=2}^{n} 1_{T_1 \cap T_i \neq \emptyset}$ the number of intersections of T_1 with $T_2, \ldots, T_n$, where 1_P denotes the indicator function of the predicate P. Then

$$\frac{\Delta}{25} \cdot E[L_1]^2(n-1) \quad \leq \quad E[I] \quad \leq \quad \Delta \cdot E[L_1]^2(n-1), \tag{3.1}$$

holds, where $\Delta := \sum_{i,j=1}^{d} p_i p_j |\sin(\delta_i - \delta_j)|$. Furthermore, for $\Delta > 0$ we obtain

$$E[I] \in \Theta\left(nE[L_1]^2\right). \tag{3.2}$$

(Distributing $L_1, \ldots, L_n$ on the interval $[0, 0.2]$ is just a technicality that simplifies the following proof of the theorem.)

Proof. We assume that $p_i > 0$ for all $1 \leq i \leq n$. Hence, Δ is zero if and only if $\delta_i - \delta_j \in \pi\mathbb{Z}$ for all $1 \leq i, j \leq d$. The latter condition means that the supporting lines of the bounded rays are parallel. Hence, two rays intersect with probability zero and the claim of the theorem is trivial. So let us assume $\Delta > 0$. The law of total expectation yields

$$E[I] = \sum_{i=1}^{d} P(\varphi_1 = \delta_i)\, E[I \mid \varphi_1 = \delta_i] = \sum_{i=1}^{d} p_i\, E[I \mid \varphi_1 = \delta_i]. \tag{3.3}$$

Consider the ray T_1 fixed. In order to have a ray T_k intersect T_1 the start point v_k of T_k needs to start in a certain area, whose shape depends on the direction φ_k of T_k. For some arbitrary direction φ we denote this area by a parallelogram S_φ that is hinged at v_1, see Figure 42:

$$S_\varphi := \left\{ v_1 + a \begin{pmatrix} \cos\varphi_1 \\ \sin\varphi_1 \end{pmatrix} - b \begin{pmatrix} \cos\varphi \\ \sin\varphi \end{pmatrix} \;:\; a \in [0, L_1], b \in [0, 0.2] \right\}.$$

The mapping

$$F : \mathbb{R}^2 \to \mathbb{R}^2 : v \mapsto \begin{pmatrix} \cos\varphi_1 & \sin\varphi_1 \\ -\sin\varphi_1 & \cos\varphi_1 \end{pmatrix} \cdot (v - v_1)$$

models the translation of v_1 to the origin and subsequent clockwise rotation by the angle φ_1. Hence, $F(T_1)$ starts at the origin and points rightwards, see Figure 42.

For a ray T_k to intersect T_1, it is necessary that the start point v_k lies is in S_{φ_k}. (Keep in mind that the lengths are restricted to $[0, 0.2]$ and T_k has the direction angle φ_k.) We denote by $(x_i', y_i') := v_i' := Fv_i$ the translated start points v_i, with $1 \leq i \leq n$. Note that T_k intersects T_1 if and only if $F(T_k)$ intersects $F(T_1)$. However, $F(T_k)$ intersects $F(T_1)$ if and only if $v_k' \in F(S_{\varphi_k})$ and $L_k |\sin(\varphi_k - \varphi_1)| \geq |y_k'|$. (The latter condition basically states that T_k is long enough in order to intersect T_1.)

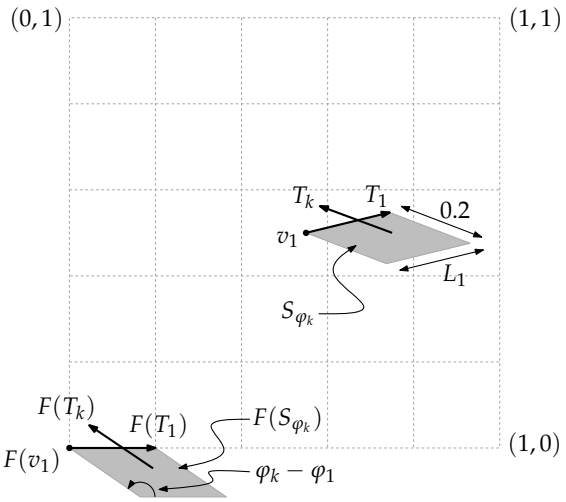


Figure 42: The big 5×5 grid illustrates $[0,1]^2$ with the origin at the left bottom point. We see that $\lambda(S_{\varphi_k}) = L_1 \cdot 0.2 \cdot |\sin(\varphi_k - \varphi_1)|$, where λ denotes the Lebesgue measure.

We note that $S_\varphi \subset [0,1]^2$ holds for all $\varphi_1 \in D$ and $\varphi \in [0, 2\pi)$ only if $v_1 \in [0.4, 0.6]^2$. Hence, for any $\delta_i \in D$ it follows that

$$E[I \mid \varphi_1 = \delta_i] \leq E[I \mid \varphi_1 = \delta_i, v_1 \in [0.4, 0.6]^2].$$

On the other hand, by the law of total expectation we obtain

$$P(v_1 \in [0.4, 0.6]^2) \cdot E[I \mid \varphi_1 = \delta_i, v_1 \in [0.4, 0.6]^2] \leq E[I \mid \varphi_1 = \delta_i],$$

and therefore

$$\frac{1}{25} \cdot E[I \mid A_i] \;\; \leq \;\; E[I \mid \varphi_1 = \delta_i] \;\; \leq \;\; 1 \cdot E[I \mid A_i],$$

where A_i denotes the event that $\varphi_1 = \delta_i$ and $v_1 \in [0.4, 0.6]^2$. Summing up over all i according to Equation (3.3) gives

$$\frac{1}{25} \cdot \sum_{i=1}^{d} p_i E[I|A_i] \;\; \leq \;\; E[I] \;\; \leq \;\; 1 \cdot \sum_{i=1}^{d} p_i E[I|A_i]. \tag{3.4}$$

In the next step, we analyze $E[I|A_i]$. Let I_j denote the number of intersections caused by rays having a direction angle δ_j. Hence $\sum_{j=1}^{d} I_j = I$. We further denote by $B_{i,j,m} \subseteq A_i$ those events of A_i, where exactly m rays point to direction δ_j. Note that each ray causes

intersections independently to each other. Therefore, we can separate the cases according to the distribution of the direction angles $\varphi_2, \dots, \varphi_n$, which is multinomial:

$$\begin{aligned} E[I|A_i] &= \sum_{j=1}^{d} E[I_j|A_i] \\ &= \sum_{j=1}^{d} 25 \cdot \int_{[0.4,0.6]^2} \int_0^{0.2} \sum_{n_1+\dots+n_d=n-1} \\ &\quad \binom{n-1}{n_1,\dots,n_d} p_1^{n_1} \cdots p_d^{n_d} E[I_j|B_{i,j,n_j}] \, \mathrm{d}f(L_1)\mathrm{d}v_1. \end{aligned} \tag{3.5}$$

Next, we analyze $E[I_j|B_{i,j,m}]$. We observe that $E[I_j|A_i]$ is zero for $i = j$. Hence, we may assume $i \neq j$ in the sequel. Recall that we are asking for the expected number of intersections of T_1 with m rays that point in direction δ_j and which are distributed independently. Hence, we may assume that the rays $T_2, \dots, T_{m+1}$ are driving in direction δ_j. Recall that T_k intersects T_1 only if $v_k \in S_{\delta_j}$. Denoting by λ the Lebesgue measure, we obtain

$$E[I_j|B_{i,j,m}] = \sum_{l=0}^{m} \binom{m}{l} \lambda(S_{\delta_j})^l (1 - \lambda(S_{\delta_j}))^{m-l} E[I_j|A_{i,j,l}],$$

where $A_{i,j,l} \subseteq B_{i,j,m}$ denotes the event that exactly l of the rays of $B_{i,j,m}$ start within S_{δ_j}.

We now resolve $E[I_j|A_{i,j,l}]$. W.l.o.g. assume that $T_2, \dots, T_{l+1}$ start in S_{δ_j}. Recall the notation $(x'_k, y'_k) := v'_k := F(v_k)$ and recall that we may assume $i \neq j$, since $E[I_j|A_i]$ is zero for $i = j$. Since every ray causes intersections independently, we get

$$\begin{aligned} E[I_j|A_{i,j,l}] &= \sum_{k=2}^{l+1} \frac{1}{\lambda(S_{\delta_j})} \int_{S_{\delta_j}} \int_0^{0.2} 1_{T_k \cap T_1 \neq \emptyset} \, \mathrm{d}f(L_k)\mathrm{d}v_k \\ &= l \frac{1}{\lambda(S_{\delta_j})} \int_{FS_{\delta_j}} \int_0^{0.2} 1_{L_2|\sin(\delta_i-\delta_j)| \geq |y'_2|} \, \mathrm{d}f(L_2)\mathrm{d}v'_2 \\ &= l \frac{1}{\lambda(S_{\delta_j})} L_1 \int_0^{0.2|\sin(\delta_i-\delta_j)|} P(L_2|\sin(\delta_i - \delta_j)| \geq y'_2) \, \mathrm{d}y'_2. \end{aligned}$$

Next, we substitute y'_2 by $z := y'_2 / |\sin(\delta_j - \delta_i)|$ and get

$$\begin{aligned} E[I_j|A_{i,j,l}] &= l \frac{|\sin(\delta_j - \delta_i)|}{\lambda(S_{\delta_j})} L_1 \int_0^{0.2} P(L_2 \geq z) \, \mathrm{d}z \\ &= l \frac{|\sin(\delta_j - \delta_i)| L_1}{\lambda(S_{\delta_j})} E[L_2] \\ &= 5lE[L_1]. \end{aligned} \tag{3.6}$$

Since $\sum_{k=0}^{n} \binom{n}{k} p^k (1-p)^{n-k}$ equals 1, we can plug the last result into the expression for $E[I_j|B_{i,j,m}]$ and get

$$\begin{aligned} E[I_j|B_{i,j,m}] &= \sum_{l=0}^{m} \binom{m}{l} \lambda(S_{\delta_j})^l (1-\lambda(S_{\delta_j}))^{m-l} 5l E[L_1] \\ &= 5E[L_1]\lambda(S_{\delta_j}) m \cdot \sum_{l=1}^{m} \binom{m-1}{l-1} \lambda(S_{\delta_j})^{l-1} (1-\lambda(S_{\delta_j}))^{m-1-(l-1)} \\ &= 5E[L_1]\lambda(S_{\delta_j}) m \\ &= mE[L_1]L_1|\sin(\delta_i - \delta_j)|. \end{aligned}$$

In the final step, we plug this result into Equation (3.5) and obtain

$$\begin{aligned} E[I|A_i] = \sum_{j=1}^{d} 25 \int_{[0.4,0.6]^2} \int_0^{0.2} & \sum_{n_1+\cdots+n_d=n-1} \binom{n-1}{n_1,\ldots,n_d} p_1^{n_1} \cdots p_d^{n_d} \\ & n_j E[L_1] L_1 |\sin(\delta_i - \delta_j)| \, \mathrm{d}f(L_1) \, \mathrm{d}v_1 \\ = 25 \int_{[0.4,0.6]^2} \mathrm{d}v_1 \cdot E[L_1] & \int_0^{0.2} L_1 \, \mathrm{d}f(L_1) \cdot \sum_{j=1}^{d} |\sin(\delta_i - \delta_j)| \\ & \sum_{n_1+\cdots+n_d=n-1} n_j \binom{n-1}{n_1,\ldots,n_d} p_1^{n_1} \cdots p_d^{n_d}. \end{aligned} \tag{3.7}$$

Next we use

$$n_j \binom{n-1}{n_1,\ldots,n_d} = (n-1) \binom{n-2}{n_1,\ldots,n_j-1,\ldots n_d}$$

and therefore see that

$$\sum_{n_1+\cdots+n_d=n-1} n_j \binom{n-1}{n_1,\ldots,n_d} p_1^{n_1} \cdots p_d^{n_d} \quad = \quad (n-1)p_j.$$

Finally, by $E[L_1] = \int_0^{0.2} L_1 \, \mathrm{d}f(L_1)$, it follows that

$$E[I|A_i] = E[L_1]^2 (n-1) \cdot \sum_{j=1}^{d} p_j |\sin(\delta_i - \delta_j)|. \tag{3.8}$$

Using this result in Equation (3.4) finally proves the assertions of the theorem. □

Choosing the lengths $L_1, \ldots, L_n$ on the interval $[0, 0.2]$ was a technical twist. It allowed us to assume that if a bounded ray T_1 starts in $[0.4, 0.6]^2$ and is intersected by another bounded ray T_2 then T_2 started definitely in $[0,1]^2$, because the start points of T_1 and T_2 have a distance of at most 0.4. Therefore, it holds that

$$\frac{1}{25} \cdot E[I|A] \leq E[I] \leq E[I|A],$$

where A denotes the event that T_1 started in $[0.4, 0.6]^2$. Note that the left inequality follows by the law of total expectation. In order to prove Theorem 3.3 it remained to show that $E[I|A] = \Delta \cdot E[L_1]^2(n-1)$. However, if we distribute $L_1, \ldots, L_n$ on an interval $[0, \epsilon]$, with any positive $\epsilon < 0.25$, the argument from above easily extends to

$$(1-4\epsilon)^2 \cdot E[I|A] \leq E[I] \leq E[I|A],$$

where A denotes the event that $v_1 \in [2\epsilon, 1-2\epsilon]^2$. Note that the remainder of the proof of Theorem 3.3 is not affected by the above generalization. As a consequence, Theorem 3.3 can actually be generalized to

$$(1-4\epsilon)^2 \Delta \cdot E[L_1]^2(n-1) \leq E[I] \leq \Delta \cdot E[L_1]^2(n-1). \tag{3.9}$$

3.2.2 Implications to the motorcycle graph

Consider n random motorcycles $m_1, \ldots, m_n$ that drive at unit speed and where the start points $v_1, \ldots, v_n$ are chosen uniformly from the unit square $[0,1]^2$ and the direction angles $\varphi_1, \ldots, \varphi_n$ are chosen from a set $D := \{\delta_1, \ldots, \delta_d\}$, where $\delta_i \in [0, 2\pi)$ occurs with probability p_i, as in Theorem 3.3. After generating n random motorcycles as described above, we compute the motorcycle graph $\mathcal{M}(m_1, \ldots, m_n)$ and record all the trace lengths. We can repeat this experiment a number of times and keep on recording the trace lengths. The samples recorded can be used to obtain an approximation $\hat{f}$ of the density of the trace lengths of a motorcycle graph with n motorcycles. Using the approximate density $\hat{f}$ in Theorem 3.3 establishes the relation (3.9) between the expected number of intersections $E[I]$ and the mean trace length $E[L_1]$ according to the approximate density function $\hat{f}$. Unfortunately, both $E[I]$ and $E[L_1]$ are unknown.

However, for increasingly larger values of n the vast majority of motorcycles does not reach the boundary of $[0,1]^2$, but crashes against other traces. Hence, as the number n of motorcycles increases, the trace lengths shrink in the average case[2], and for sufficiently large n the vast majority of motorcycles can be expected to have a trace length less than some constant ϵ smaller than 0.25. Since there are at most n crashes, a motorcycle trace may be assumed to intersect two other traces on average: the motorcycle itself crashes into another trace and a second motorcycle crashes into the considered trace. This suggests $E[I] = 2$, which can also be verified experimentally. (Actually, $E[I] \in O(1)$ would suffice for our subsequent runtime analysis of our algorithm in Section 3.3.)

Plugging $E[I] = 2$ and a small ϵ in the inequality (3.9) suggests the following approximation for the mean trace length:

$$E[L_1] \approx \sqrt{\frac{2}{(n-1)\sum_{i,j=1}^{d} p_i p_j |\sin(\delta_i - \delta_j)|}}. \tag{3.10}$$

Of course, the assumption that $L_1, \ldots, L_n$ are independently distributed is not justified for the actual motorcycle graph problem. However, in Section 3.3.3, we are able to substantiate this approximative formula for the mean trace length of motorcycle graphs by providing sound experimental evidence.

2 Of course, the motorcycles must not all drive in parallel directions.

3.3 A SIMPLE AND PRACTICE-MINDED IMPLEMENTATION

In order to compute the straight skeleton $\mathcal{S}(G)$ of a planar straight-line graph G by using Bone, it is vital that we can compute the motorcycle graph $\mathcal{M}(G)$ fast in practice. The algorithm by Eppstein and Erickson [EE99] would allow us to compute $\mathcal{M}(G)$, but the algorithm is too complicated to be implemented. The algorithm by Cheng and Vigneron [CV07] is easier, but still complicated to implement — e. g., we would need to implement an algorithm for the $1/\sqrt{n}$-cutting — and it needs to know all motorcycles a priori. The standard priority-queue based algorithm is trivial to implement, but exhibits an $O(n^2 \log n)$ runtime in practice. For our purposes, we seek a motorcycle graph implementation that is simple enough to be implemented, runs fast in practice, and which supports the dynamic insertion of new motorcycles.

Our approach is to take the algorithm of Cheng and Vigneron [CV07], drop the arrangements on each cutting cell and to replace the $1/\sqrt{n}$-cutting by a regular $\sqrt{n} \times \sqrt{n}$ grid. In other words, we apply geometric hashing to the standard priority-queue based algorithm. The motivation for our approach is a simple trade-off: we lose the deterministic $O(n\sqrt{n} \log n)$ time complexity, but instead gain an algorithm that (i) is easy to implement and (ii) runs in $O(n \log n)$ time if the motorcycles are sufficiently uniformly distributed.

3.3.1 Details of the algorithm

The input to our algorithm consists of a set $M = \{m_1, \ldots, m_n\}$ of motorcycles and a set $W = \{w_1, \ldots, w_u\}$ of rigid walls. The motorcycles need not all be known a-priori. A wall is modeled as a straight-line segment and a motorcycle m_i is given by a start point v_i, a speed vector s_i and a start time $t_i^* \in [0, \infty)$. Due to practical numerical advantages, we scale the input such that the bounding box of the start points is a proper subset of the unit square $[0, 1]^2$. For the matter of simplicity, we first restrict our computation of the motorcycle graph to the unit square $[0, 1]^2$. However, this restriction can be waived easily, see Section 3.3.4. (The restriction of the computation to $[0, 1]^2$ can be enforced within our framework by adding four dummy walls that form the boundary of $[0, 1]^2$.)

Our algorithm maintains two geometric hashes, H_M and H_W, which form regular $\sqrt{n} \times \sqrt{n}$ grids on $[0, 1]^2$. While H_M keeps track of the motorcycles, H_W contains the walls of W. We use H_W in order to determine whether a motorcycle crashed against a wall. However, since we consider walls to be rigid, we could have employed more efficient ray-shooting algorithms in terms of worst-case complexity. For the matter of simplicity, we choose the geometric hash for our implementation.

The basic algorithm is a discrete event simulation of the movement of the motorcycles with two types of events: crash events and switch events. A crash event indicates that a motorcycle crashes against another motorcycle or a wall, and a switch event occurs when a motorcycle leaves one grid cell and enters a neighboring one. All events pending are kept in a priority queue Q. Furthermore, for every motorcycle m_i, we maintain a balanced binary search tree $C[m_i]$ that contains potential future crash events of the motorcycle m_i.

The algorithm starts with filling H_W with all walls of W and then invokes `insertMc(m)` for each $m \in M$, which inserts a new motorcycle m to our data structures. The main loop

of the algorithm extracts one event e from Q after the other and processes them by calling `handle(`e`)`, depending on the actual type of the event e. If a newly emerging motorcycle m should be inserted at any time of computation then `insertMc(`m`)` is called. The procedures `insertMc` and `handle` are described in the sequel:

- **`insertMc(motorcycle` m`)`**: We first create an empty binary search tree $C[m]$ and then insert a switch event for m into Q. The occurrence time of the switch event is set to the start time of m.
- **`handle(switch event` e **`of the motorcycle` m`)`**: We register m at the cell that m entered and add the subsequent switch event of m to Q, if one exists. Then we check for a potential crash against a wall in the current cell and add the earliest one, if existing, as a crash event to Q. We clear $C[m]$ and for every other motorcycle m' registered in the current cell, we check for an intersection of the tracks of m and m'. For each such intersection, we add a corresponding crash event into $C[m]$ if m' reaches the intersection before m, and into $C[m']$ for the dual case. Note that if we add an event into $C[m']$ that ends up being the earliest in $C[m']$ then we have to update Q accordingly. Finally, we add the earliest crash event of $C[m]$ into Q.
- **`handle(crash event` e **`of the motorcycle` m`)`**: First, we mark the motorcycle m as crashed and clear $C[m]$. Note that the trace of m ends at the corresponding crash point. Secondly, we remove the remaining switch event of m from Q. (Alternatively, we could leave the event in Q, but check at each switch event whether the current event is still valid.) Then we clean up interactions with other motorcycles m' in the current grid cell: We remove from Q all crash events, where m is involved and which got invalid, because it turned out that m will not reach the location of the potential crash event. Likewise, if $C[m']$ contains such an invalid crash event then it is removed as well.

3.3.2 Runtime analysis

In the subsequent analysis, we ignore the influence of the wall handling and concentrate on the computation of the motorcycle graph only. The procedure `insertMc` is called exactly n times, which takes $O(n \log n)$ time in total. A single crash event or switch event of a motorcycle is handled in $O(k \log n)$ time, where $k \in O(n)$ denotes the number of motorcycles in the cell affected. Note that we can remove an element of Q in $O(\log n)$ time if we have a pointer to the element and if we use, for instance, a maximizing heap to implement Q.

In total we have $O(n)$ crash events and at most $O(n\sqrt{n})$ switch events. Hence, in the worst case, our algorithm runs in $O(n^2\sqrt{n} \log n)$ time. However, it seems very unlikely that the worst case actually happens: it would require that $\Omega(n)$ motorcycles drive across $\Omega(\sqrt{n})$ common grid cells. Hence, those $\Omega(n)$ motorcycles drive virtually parallel along a long strip that is only $O(1/\sqrt{n})$ units thick and, moreover, no other motorcycle is allowed to cross this strip, until the motorcycles crossed a constant fraction of the whole grid.

As we learned in Section 3.2, a motorcycle is visiting $O(1)$ grid cells on average, if the motorcycles are distributed uniformly enough. Consequently, a single grid cell is occupied by $O(1)$ motorcycles on average. Again, the initialization consumes $O(n \log n)$ time in total. However, now a single crash event or switch event is handled in $O(\log n)$ time in the average case. Still, we have $O(n)$ crash events but in the mean observe only $O(1)$ switch

events per motorcycle. Summarizing, we may expect a runtime of $O(n \log n)$ for sufficiently uniformly distributed input, as motivated by Section 3.2. We provide sound experimental evidence in Section 3.3.3 that underpins our arguments for an $O(n \log n)$ runtime for uniformly distributed start points, but also demonstrates an $O(n \log n)$ runtime for most of the real-world input, where start points are not necessarily distributed uniformly.

3.3.3 Experimental results and runtime statistics

Our motorcycle code is called MOCA[3]. It is implemented in C++ and uses the STL for common data structures like queues, red-black trees and priority queues. All geometric computations are based on ordinary IEEE 745 double-precision floating-point arithmetic. MOCA provides runtime options for a posteriori tests to check necessary conditions for the correctness of the resulting motorcycle graph. In particular, we check (i) for motorcycle traces with a free end[4] and (ii) for motorcycle traces intersecting in the relative interiors of each other. To the best of our knowledge, this is the first competitive motorcycle graph implementation. For this reason, we do not compare our code with other implementations, but content ourselves with a discussion on the performance of MOCA.

3.3.3.1 *Verification of the stochastic analysis*

We used MOCA to collect statistical properties of several datasets, in order to underpin the theoretical results obtained in Section 3.2. We set up three experiments that investigate the dependence of the mean trace length on (i) the number of motorcycles n and (ii) the direction angles δ_i in Equation (3.10).

For the first two experiments, we created datasets with n random motorcycles by choosing the start points uniformly in $[0,1]^2$ and the direction angle uniformly from the set $\{0, \delta\}$. Equation (3.10) asserts that the mean trace length L is given by

$$E[L] \approx \frac{2}{\sqrt{(n-1)|\sin\delta|}}. \tag{3.11}$$

In Experiment 1, we created a dataset for each $n \in \{i \cdot 5\,000 \;:\; 1 \le i \le 60\}$ and $\delta \in \{i\pi/12 \;:\; 1 \le i \le 6\}$ and used MOCA to determine the mean trace length among each dataset. In the left subfigure of Figure 43, each dot depicts the mean trace length of a dataset, where the resulting values were normalized for illustrative reasons, by dividing them by the factor $2/\sqrt{n-1}$. As predicted by Equation 3.11, the plot shows six[5] horizontal lines, where each line corresponds to a particular value of δ. In Experiment 2, we created datasets for $n = 10\,000$ and $\delta \in \{i\pi/40 \;:\; 1 \le i \le 40\}$. The reciprocal values of the normalized mean trace lengths are shown in the right subfigure of Figure 43. The normalized mean trace lengths are aligned on the reference curve $\sqrt{|\sin\delta|}$, which matches the estimation provided by Equation (3.11).

3 Short-hand for MOtorcycle CrAsher.
4 Each endpoint of a trace of a motorcycle that did not escape must coincide with the trace of another motorcycle.
5 The two bottom lines mostly overlap.

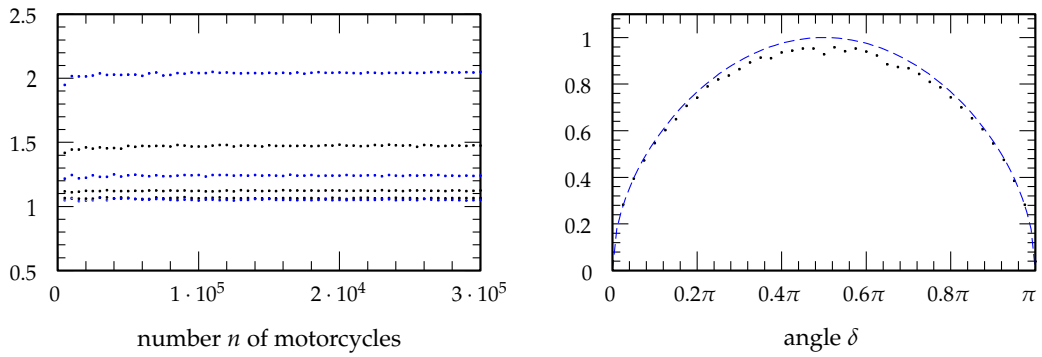


Figure 43: Two experiments illustrating the mean trace length of n motorcycles. A dataset contains motorcycles with uniformly distributed start points and uniformly distributed directions on the set $\{0, \delta\}$. Left, Experiment 1: every dot depicts the mean trace length for different n and δ. The resulting values are divided by $2/\sqrt{n-1}$. Right, Experiment 2: every dot depicts the reciprocal of the mean trace length for a fixed n. The x-axis illustrates δ. The reference curve $\sqrt{|\sin \delta|}$ is shown in blue.

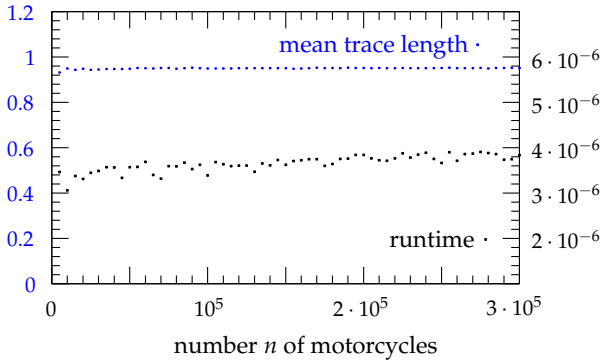


Figure 44: The runtime of Moca and the mean trace length of random dataset containing n motorcycles. The start points are uniformly distributed on $[0, 1]^2$ and the directions are uniformly distributed on $[0, 2\pi)$. Each blue dot depicts the mean trace length and each black dot the runtime on a single dataset. For illustrative reasons we divided the runtime by $n \log n$ and the mean trace length by $\sqrt{\frac{\pi}{n-1}}$.

In the third experiment, we considered the mean trace length of datasets where the direction angles are distributed uniformly on $[0, 2\pi)$. This can be achieved by uniformly distributing the direction angles on $\{\delta_1, \ldots, \delta_d\}$, with $\delta_i = i\frac{2\pi}{d}$, and subsequently considering $d \to \infty$. According to Equation (3.10) we obtain:

$$\begin{aligned} E[L] &\approx \lim_{d\to\infty} d\sqrt{\frac{1}{(n-1)\sum_{i=1}^{d-1}(d-i)|\sin i\frac{2\pi}{d}|}} \\ &= \lim_{d\to\infty} d\sqrt{\frac{1}{(n-1)\cdot d^2\sum_{i=1}^{d-1}(1-\frac{i}{d})|\sin 2\pi\frac{i}{d}|\cdot\frac{1}{d}}} \\ &= \sqrt{\frac{1}{(n-1)\int_0^1(1-x)|\sin 2\pi x|\,dx}} \\ &= \sqrt{\frac{\pi}{n-1}}. \end{aligned} \tag{3.12}$$

We again generated random datasets, where n ranges from 5 000 to 300 000 in steps of 5 000. Figure 44 shows the runtimes and the mean trace lengths on these datasets. The runtime has been divided by $n \log n$ and the mean trace lengths have been normalized by the factor $\sqrt{\pi/n-1}$. As predicted, the graph shows two horizontal lines. That is, the runtime of Moca is in $O(n \log n)$ and the mean trace length is approximately $\sqrt{\pi/n-1}$.

3.3.3.2 *Runtime statistics on real world data*

We performed the following runtime tests on a Linux machine with a 32-bit Kernel. We used an Intel E6700 Core 2 Duo processor, clocked at 2.66 GHz, using 4 GiB of memory. Note that the 32-bit architecture limits the memory footprint of Moca to roughly 3 GiB in user space. For time measurement, we used the C library function `getrusage`.

Since the development of Moca is motivated by our straight-skeleton implementation Bone, we consider planar straight-line graphs G as input and test Moca by computing the motorcycle graph $\mathcal{M}(G)$ induced by G. We ran Moca on more than 22 000 datasets, consisting of synthetic and real-world data. Our real-world datasets include polygonal cross-sections of human organs, GIS maps of roads and river networks, polygonal outlines of fonts, and boundaries of work-pieces for CNC machining or stereo-lithography. The synthetic test data was generated by means of RPG [AH96] and an enhanced version of RPG due to Held. The synthetic data also contains contrived data, like extremely fine approximations of smooth curves, where the vertices are distributed highly irregularly.

Figure 45 (a) illustrates the runtime of Moca on each dataset. The runtimes are given in seconds and were divided by $n \log n$ for illustrative reasons. To avoid unreliable timings and other idiosyncrasies of small datasets, we only plot the results of datasets with at least 100 motorcycles. We observe that Moca exhibits a runtime of 2 to $10 \cdot n \log n$ µs for the vast majority of our datasets. About 100 outliers that did not fit into this plot, took up to $2 \cdot n \log n$ ms. A typical dataset of this kind is a sampled ellipse: all motorcycles escape and visit a large number of cells.

The plot in Figure 45 (b) shows the mean trace length, which has been multiplied by $\sqrt{n}$ for a better illustration. We can see that for most datasets in the entire database, the mean

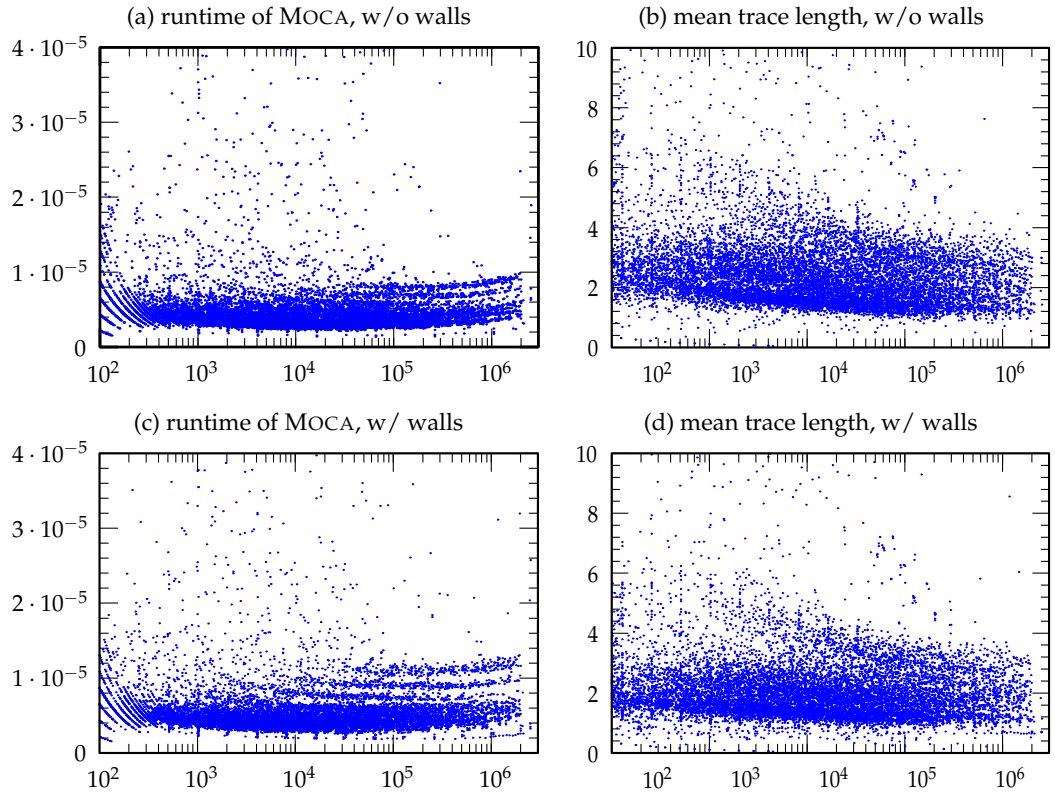


Figure 45: A dot depicts the runtime of MOCA resp. the mean trace length of a dataset. The x-axis show the size n of the datasets. (a) Runtime of MOCA in seconds, divided by $n \log n$. (b) Mean trace length, multiplied with $\sqrt{n}$. (c, d) Show the plots analogous to (a, b), but with walls inserted.

trace length is between $0.5/\sqrt{n}$ and $6/\sqrt{n}$. This circumstance is the main reason for the good runtime behavior achieved by our implementation. Our theoretical analysis carried out in Section 3.2 is based on the assumption that the start points are distributed uniformly in the unit square. Figure 45 (b) provides experimental evidence that this assumption can be relaxed for real-world input, which still exhibits an average trace length of $O(\sqrt{n})$ for most datasets.

For the test runs of subfigure (a) and (b), we did not insert the edges of the input graphs G as walls. However, additional tests demonstrate that inserting the walls has only a negligible impact on the runtime of MOCA resp. the mean trace length of the resulting motorcycle graphs. We illustrated the corresponding runtimes and mean trace lengths in subfigure (c) and (d).

We further investigated the runtime of MOCA on random datasets with non-uniformly distributed start points. For this reason, we generated datasets where the start points resp. the x-coordinates of the start points are distributed Gaussian resp. multi-modal Gaussian. By varying the standard deviations, we are able to successively concentrate the start points

at specific regions of the unit square. In Figure 46, we plotted the runtime of Moca and the mean trace lengths on datasets, where (a) the start points are distributed Gaussian at the center of the unit square and (b) only the x-coordinates are distributed Gaussian with mean 0.5. As expected, for a decreasing standard deviation the runtime of Moca increases accordingly. Note that the mean trace length decreases as well, but the impact on the runtime due to the condensed start points dominates the result.

3.3.4 Extending the computation beyond the unit square

The computation of the motorcycle graph outside the unit square can be done very efficiently in terms of worst-case complexities. At first, we compute the motorcycle graph inside the unit square. When a motorcycle reaches the boundary of the unit square it is temporarily stopped. After all motorcycles crashed or have been stopped at the boundary of the unit square, we compute the motorcycle graph outside the unit square. This, however, can be done very efficiently: We interpret the boundary of the growing unit square as a sweep-line. Each motorcycle sits on this growing square and whenever two motorcycles m_1, m_2 would exchange their positions — i. e., the square reached an intersection point of the tracks of m_1 and m_2 — then either m_1 crashed into m_2 or vice versa. The crashed motorcycles are removed from the sweep line and the algorithm continues. The algorithm is correct because the motorcycles are, roughly speaking, moving in the same direction as the wavefront and never against it. Using a balanced binary tree structure on the sweep line allows us to compute the motorcycle graph outside the unit square in $O(n \log n)$ time.

This strategy is motivated by the sweep-line algorithm due Eppstein and Erickson [EE99]. Their algorithm computes the motorcycle graph of motorcycles, where the velocities have positive x-coordinates, in $O(n \log n)$ time. Note that our approach also works if we use the convex hull of the start points instead of the boundary of the unit square as initial sweep line. In other words, the motorcycle graph outside the convex hull of the start points can be computed in $O(n \log n)$ time. However, the computation of the motorcycle graph within the convex hull remains complicated.

Note that the approach presented above assumes that no motorcycle emerges during the propagation of the sweep line. However, under specific circumstances we can still allow the launch of new motorcycles: (i) the start point needs to coincide with the current position of the wavefront and (ii) the motorcycle needs to drive to the one side of the wavefront that has not yet been swept. Both conditions are fulfilled by our generalization of the motorcycle graph.

Moca pursues a simple approach to continue the computation of the motorcycle graph outside the unit square, by extending the $2(\sqrt{n}+1)$ grid lines to infinity. We consider the resulting infinite grid cells as part of the hash and continue the simulation of the moving motorcycles on this grid cells. In order to restrict our computations to the unit square, we add four dummy walls that cover the boundary of the unit square. The impact on the performance of Moca, when the boundary walls are removed, directly depends on the number of motorcycles that do not crash within the unit square, which includes the motorcycles that escape.

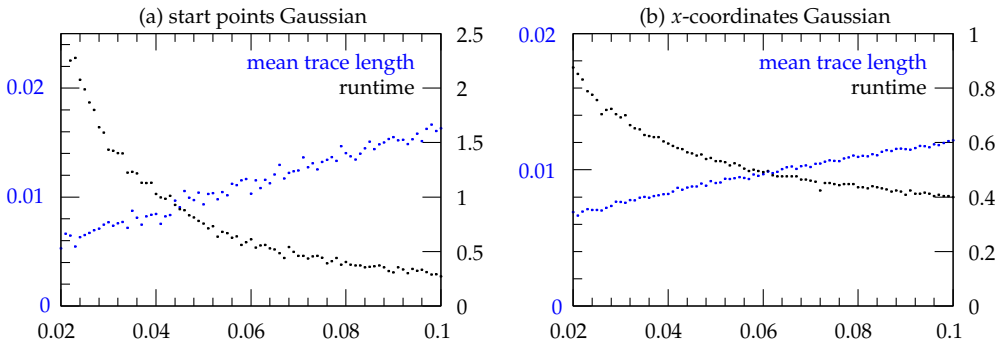


Figure 46: The runtime of Moca and the mean trace length on random datasets, where the direction angles are distributed uniformly on $[0, 2\pi)$. (a) 2 000 motorcycles, where the start points are distributed Gaussian with mean $(0.5, 0.5)$. (b) 10 000 motorcycles, where the x-coordinates of the start points are distributed Gaussian with mean 0.5 and the y-coordinates are distributed uniformly on $[0, 1]$. In both subfigures, the x-axis shows the corresponding standard deviation.

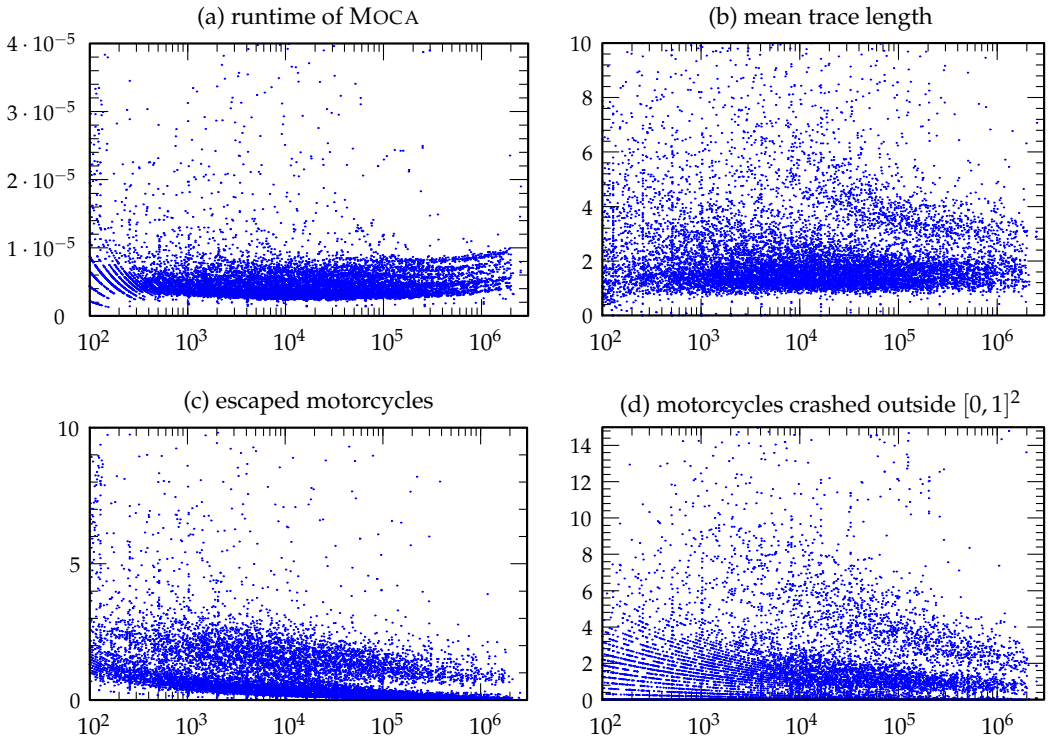


Figure 47: Statistics obtained from Moca when the computation is continued beyond the unit square. The x-axis depicts the number n of motorcycles in the dataset. (a) Runtime, divided by $n \log n$. (b) Mean trace length of the crashed motorcycles, multiplied with $\sqrt{n}$. (c) Number of escaped motorcycles, divided by $\sqrt{n}$. (d) Number of motorcycles crashed outside the unit square, divided by $\sqrt[4]{n}$.

Figure 47 (a) and (b) show the runtime of MOCA and the mean trace length, where the computation is continued beyond the unit square. We observe that there is only a little impact on the runtime of MOCA and the same holds for the mean trace length. In subfigure (c) we plotted the number of motorcycles that escaped and subfigure (d) shows the number of motorcycles that crashed outside of the unit square. Note that the remaining motorcycles crashed within the unit square. We observe that the number of motorcycles that escaped is roughly $\Theta(\sqrt{n})$ and the number of motorcycles that crashed outside $[0,1]^2$ is approximately $\Theta(\sqrt[4]{n})$. Hence, the vast majority of motorcycles remained within the unit square for most of the datasets in our database. This is the reason why MOCA performs well, even though we continue the computation of the motorcycle graph outside the unit square.

3.4 EXTRACTING THE MOTORCYCLE GRAPH FROM THE STRAIGHT SKELETON

In the introduction of this chapter, we mentioned the close relation between straight skeletons and motorcycle graphs. From a geometric point of view, this relation becomes visible by Theorem 2.26, which states that the motorcycle graph covers the reflex arcs of the straight skeleton. In general, it can be observed that the gap between the reflex arcs and the motorcycle traces decreases as the motorcycles resp. the reflex wavefront vertices move faster. From this observation, the question arises whether we can always approximate the motorcycle graph using the straight skeleton. In other words, can we compute the motorcycle graph using a straight-skeleton algorithm?

We think that this question is interesting due to following reasons. Firstly, since motorcycle graphs play an important role in the computation of straight skeletons, it appears to be important to deepen the insight into the geometric relation between motorcycle graphs and straight skeletons. Secondly, if we manage to efficiently reduce the construction problem of motorcycle graphs to straight skeletons then we are able to describe the complexity of one problem in terms of the other problem. In fact, in Section 3.4.3, we present a proof for the P-completeness of straight skeletons that is based on the results of the subsequent sections and the P-completeness of motorcycle graphs due to Eppstein and Erickson [EE99].

3.4.1 Approximating the motorcycle graph by the straight skeleton

We assume that n motorcycles $m_1, \ldots, m_n$ are given, where each motorcycle m_i has a start point p_i and a speed vector v_i. All motorcycles start at the same time. In particular, no motorcycles emerge later on. Can we find an appropriate planar straight-line graph G such that the straight skeleton $\mathcal{S}(G)$, resp. a proper subset of $\mathcal{S}(G)$, approximates $\mathcal{M}(m_1, \ldots, m_n)$ up to a given tolerance?

It follows from Theorem 2.26 that if we construct G such that at each p_i a reflex wavefront vertex starts moving along the track of m_i with velocity v_i then the reflex arcs of $\mathcal{S}(G)$ that belong to these wavefront vertices approximate the traces of $\mathcal{M}(m_1, \ldots, m_n)$ up to some gap. The simplest way in order to obtain such reflex wavefront vertices is to place isosceles

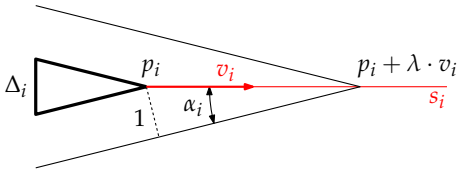


Figure 48: The isosceles triangle Δ_i is placed on the vertex p_i. The interior angle of Δ_i at p_i is $2\alpha_i$ and $\lambda|v_i| = 1/\sin\alpha_i$.

triangles Δ_i at each p_i such that the trace of m_i is bisecting the exterior angle of Δ_i. By setting the angle of Δ_i at p_i accordingly, we can adapt the speed of the corresponding wavefront vertex.

As already mentioned, we observe that the faster m moves the better is its trace approximated by the reflex straight-skeleton arc that is covered by m. But since the speed of $m_1, \ldots, m_n$ is part of the input to our problem setting, we cannot change the speeds of the motorcycles. Moreover, a wavefront vertex has always a speed of at least 1. If a motorcycle has a speed less than 1 then we cannot construct an according Δ_i, as mentioned above. However, it easy to see that if we multiply each speed vector v_i by the same constant $\lambda > 0$ then the motorcycle graph remains the same. Hence, the idea is to place at each p_i an isosceles triangle Δ_i, where the interior angle at p_i equals $2\alpha_i$ and α_i is given by

$$\alpha_i := \arcsin \frac{1}{\lambda|v_i|}. \tag{3.13}$$

The constant λ is chosen large enough such that $\lambda|v_i| > 1$ for all $1 \leq i \leq n$, see Figure 48. The size of Δ_i will be specified later. We denote by u_i the reflex wavefront vertex emanated at p_i. Note that u_i has λ-times the speed of m_i. Furthermore, note that each triangle Δ_i is emanating two additional motorcycles at the other two corners. This leads us to a refined version of our initial question: Can we find λ large enough such that the reflex arcs of $\mathcal{S}(G)$ that are emanated from p_i, approximate $\mathcal{M}(m_1, \ldots, m_n)$ up to a given tolerance?

We denote the trace of each motorcycle m_i by s_i. Recall that D_μ denotes the disk with radius μ and the center at the origin. Since all traces are closed sets, there exists $\mu > 0$ such that the Minkowski sums $s_i + D_\mu$ and $s_j + D_\mu$ are disjoint for all $1 \leq i, j \leq n$, where s_i and s_j are disjoint. For example, let μ be a third of the pairwise infimum distance among all disjoint traces. Note that two traces s_i, s_j are intersecting if and only if m_i crashed into m_j or vice versa. Further, we choose μ small enough such that $p_i + D_\mu$ is disjoint with $s_j + D_\mu$ for all $1 \leq i, j \leq n$, where p_i is disjoint to s_j. We denote by G the planar straight-line graph that consists of the triangles Δ_i, as described above, where the lengths of the arms that are incident to p_i are set to $\mu/2$.

Lemma 3.4. *The wavefronts of Δ_i stay within $s_i + D_\mu$ until time $\mu/4$ if $\lambda \geq \frac{2}{|v_i|}$.*

Proof. From $\lambda \geq 2/|v_i|$ follows that $2\alpha_i$ is at most 60°. Hence, the other two angles of Δ_i are at least 60° and the two additional motorcycles at Δ_i have a speed of at most 2. Since the start points of those motorcycles have a distance of $\mu/2$ from p_i and since they drive at most a distance of $\mu/2$ in time $\mu/4$, they stay within $p_i + D_\mu$.

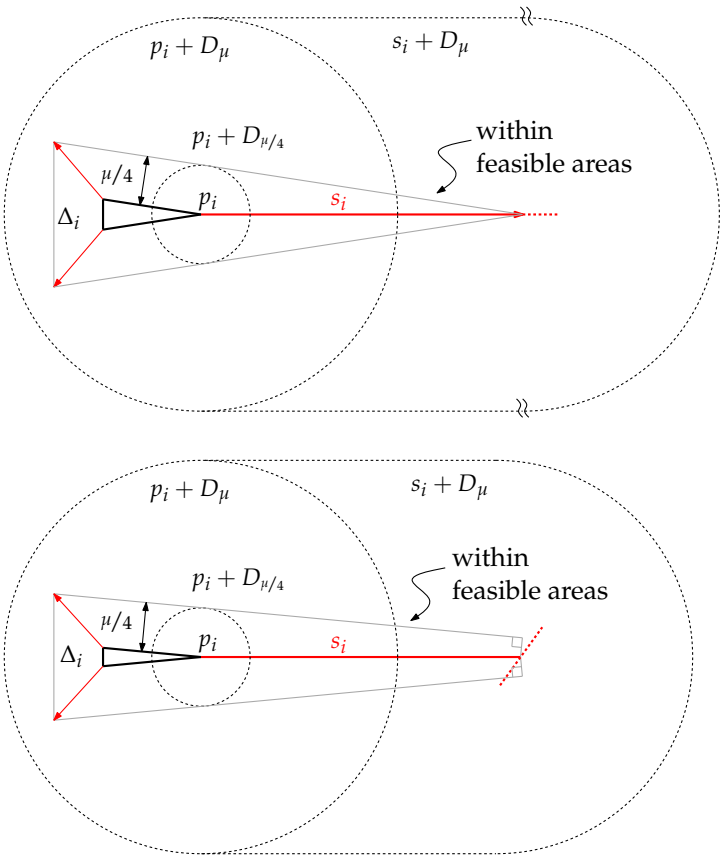


Figure 49: The wavefronts of Δ_i are bounded to $s_i + D_\mu$ until time $\mu/4$. Top: the motorcycle m_i did not crash until time $\mu/4$. Bottom: the motorcycle m_i did crash until time $\mu/4$.

Next, we consider a wavefront edge e that is emanated from Δ_i and propagating to the exterior of Δ_i. Let us recall the slabs of the lower envelope from Section 2.4.3. We call the slab that belongs to e and which is vertically projected onto the plane the *feasible area* of e. The face $f(e)$ is contained in the feasible area of e. We have to proof that $e(t)$ stays within $s_i + D_\mu$ until time $\mu/4$. First, we restrict the feasible area of e to those points that have an orthogonal distance of at most $\mu/4$ to $\overline{e(0)}$, see Figure 49. We distinguish two cases: the motorcycle m_i did crash or did not crash until time $\mu/4$. In both cases the corner points of the restricted feasible areas are contained within $s_i + D_\mu$, the restricted feasible areas are convex and $s_i + D_\mu$ is convex. Hence, the wavefronts of Δ_i until time $\mu/4$ are contained within $s_i + D_\mu$. □

We denote by $\overrightarrow{s_i}$ the ray that starts at p_i in direction v_i and define

$$L := \max_{1 \le i,j \le n} d(p_i, \overrightarrow{s_i} \cap \overrightarrow{s_j}). \tag{3.14}$$

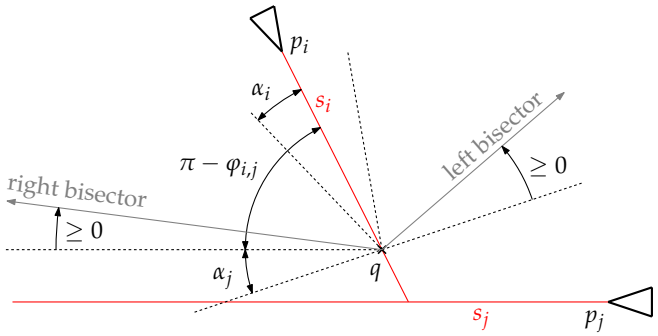


Figure 50: The wavefronts of Δ_i do not cause a crash of the reflex wavefront vertex from p_j.

Note that we may only consider indices i, j for which $\overrightarrow{s_i} \cap \overrightarrow{s_j}$ is not empty. If no such indices i, j exist then we set L to zero. Further, let us denote by $\varphi_{i,j} \in [0, \pi]$ the non-oriented angle spanned by v_i and v_j, with $\varphi_{i,j} = \varphi_{j,i}$. Next we define

$$\Phi := \min_{1 \leq i < j \leq n} \mathbb{R}^+ \cap \{\varphi_{i,j}, \pi - \varphi_{i,j}\}. \tag{3.15}$$

If the corresponding set it is empty — i.e. if all motorcycles drive on parallel tracks — then we set $\Phi := \pi/2$.

Lemma 3.5. *Let m_i denote a motorcycle that crashes into the motorcycle m_j. The wavefronts of Δ_i do not cause a split event for the reflex wavefront vertex u_j until time $\mu/4$ if $\lambda \geq 2/\min_k |v_k| \sin \Phi$.*

Proof. Since $\lambda \geq \frac{2}{|v_k| \sin \Phi}$ holds for any $1 \leq k \leq n$, it follows that

$$\sin \alpha_k \leq \frac{1}{|v_k| \lambda} \leq \frac{1}{2} \sin \Phi \leq \sin \frac{\Phi}{2},$$

because sin is concave on $[0, \pi]$. By further noting that sin is monotone on $[0, \pi/2]$ we see that

$$\alpha_k \leq \frac{\Phi}{2} \qquad \forall\, 1 \leq k \leq n.$$

The motorcycle m_j does not also crash into m_i, since two motorcycles do not crash simultaneously into each other by assumption. If s_i and s_j are collinear then the assertion is either trivial or excluded by assumption. Without loss of generality, we may assume that s_i is right of $\overrightarrow{s_j}$, see Figure 50. We denote by q the endpoint of the reflex straight-skeleton arc that is incident to p_i. Let us consider the left (resp. right) bisector between the left (resp. right) arm of m_i and the right arm of m_j, starting from q.

By Lemma 3.4 it suffices to show that the two arms of m_i do not lead to a split event with u_j until time $\mu/4$: The two additional motorcycles from Δ_i stay within $p_i + D_\mu$. Hence, we only have to consider the arms of m_i.

We conclude the proof by showing that none of both bisectors intersects $\overrightarrow{s_j}$. Let us consider the right bisector. Recall that $\alpha_i, \alpha_j \leq \Phi/2$ and that $\pi - \varphi_{i,j} \geq \Phi$. In the extremal case,

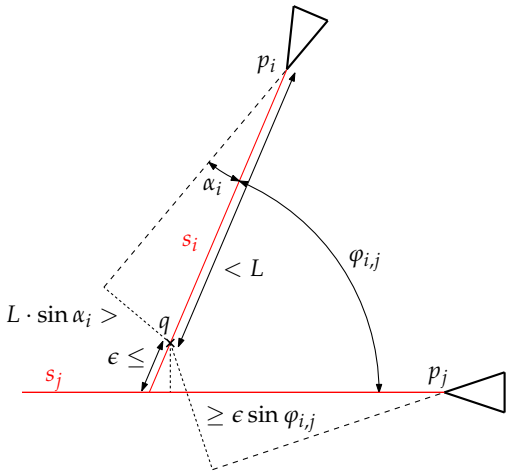


Figure 51: The point q is reached earlier by the left arm of m_i than by the right arm of m_j.

where equality is attained for all three inequalities, the right bisector is just parallel to s_j, but strictly right of $\overrightarrow{s_i}$. In all other cases the bisector rotates clockwise at q such that our assertion is true in general. Analogous arguments hold for the left bisector. Summarizing, the reflex wavefront vertex u_j does not lead to a split event with the wavefronts of Δ_i until time $\mu/4$. □

Lemma 3.6. *Let m_i denote a motorcycle crashing into the motorcycle m_j. For any $\epsilon > 0$ and*

$$\lambda \geq \frac{1}{\min_k |v_k| \cdot \sin \Phi} \cdot \max \left\{ 2, \frac{L}{\min\{\mu/4, \epsilon\}} \right\},$$

the trace s_i is covered up to a gap size ϵ by the reflex arc traced out by u_i.

Proof. We will prove the following: any point q on s_i, whose distance to the endpoint of s_i is at least ϵ, is reached by u_i no later than at time $\mu/4$, see Figure 51. We first show that until time $\mu/4$ the reflex wavefront vertex u_i may only cause a split event with the wavefronts of Δ_j. By Lemma 3.4, we know that until time $\mu/4$, the wavefronts of a triangle Δ_k could only cause a split event with u_i if s_k and s_i intersect. Hence, m_k crashed against s_i. However, by Lemma 3.5 it follows that u_i does not lead to a split event with the wavefronts from Δ_k.

W.l.o.g., we may assume that s_i lies on the right of $\overrightarrow{s_j}$. In order to show that u_i reaches q until time $\mu/4$, it suffices to prove that q has a smaller orthogonal distance to the left arm of m_i than to the right arm of m_j and that the orthogonal distance of q to the left arm of m_i is at most $\mu/4$.

The orthogonal distance of q to the left arm of m_i is at most $L \cdot \sin \alpha_i$. The orthogonal distance of q to the right arm of m_j is at least the orthogonal distance of q to s_j. However, this distance is at least $\epsilon \cdot \sin \varphi_{i,j}$. Summarizing, our assertion holds if

$$L \cdot \sin \alpha_i \leq \min\{\mu/4, \epsilon \sin \varphi_{i,j}\},$$

and thus

$$\lambda \geq \frac{L}{|v_i| \cdot \min\{\mu/4, \epsilon \sin \varphi_{i,j}\}}.$$

Our choice for λ fulfills this condition. The case where s_j and s_i are collinear such that m_i crashes at p_j is similar. The wavefront of Δ_j reaches q no later than at time $\epsilon/2$ and v_i reaches q in at most $\frac{L}{\lambda|v_i|}$ time. □

Let us denote by $\mathcal{S}^*_\lambda(m_1, \ldots, m_n) \subset \mathcal{S}(G)$ the union of the reflex straight-skeleton arcs that are traced out by $u_1, \ldots, u_n$, where G is given as described above. Then we get the following corollary of Lemma 3.6:

Corollary 3.7.

$$\lim_{\lambda \to \infty} \mathcal{S}^*_\lambda(m_1, \ldots, m_n) = \mathcal{M}(m_1, \ldots, m_n)$$

This corollary also includes that a point q on a motorcycle trace s_i of a motorcycle m_i that never crashed, is covered by an arc of $\mathcal{S}^*_\lambda(m_1, \ldots, m_n)$ for large enough λ. However, this is easy to see by applying Lemma 3.4 and Lemma 3.5, and by finally finding λ large enough such that the point q is reached by u_i until time $\mu/4$.

3.4.2 Computing the motorcycle graph

In order to reduce the construction problem of motorcycle graphs to straight skeletons, we have to cope with the remaining gaps between the motorcycle traces s_i and the reflex arcs traced out by u_i, which exist for arbitrary large λ. Hence, the question remains whether m_i actually crashed into m_j (or vice versa), even if the gap between two reflex arcs that are traced out by u_i and u_j, is very small.

In order to decide whether the motorcycle m_i escapes or whether it crashes into a trace s_j, we determine λ large enough such that the following conditions are fulfilled:

- If m_i crashes into a trace s_j then the reflex wavefront vertex u_i leads to a split event until the time $\mu/4$ and the reflex arc that is traced out by u_i has an endpoint in a straight-skeleton face of an edge of Δ_j. (The vertex u_i causes a split event with the right arm of m_j if s_i is right of $\overrightarrow{s_j}$ and the left arm if s_i is left of $\overrightarrow{s_j}$.)
- If m_i escapes then the reflex wavefront vertex u_i did not lead to a split event until the time $\mu/4$.

The following lemma states that such λ exists and provides a sufficient bound.

Lemma 3.8. *Consider $\mathcal{S}(G)$ with*

$$\lambda \geq \frac{\max\left\{2, \frac{8L}{\mu}\right\}}{\min_k |v_k| \cdot \sin \Phi}.$$

Then m_i crashes into s_j if and only if u_i leads to a split event with the wavefront emanated by Δ_j until time $\mu/4$. In particular, m_i escapes if and only if u_i does not lead to a split event until time $\mu/4$.

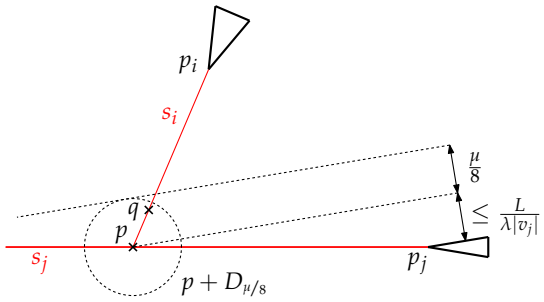


Figure 52: The reflex wavefront vertex u_i causes a split event until time $\mu/4$.

Proof. We distinguish two cases. First, suppose that the motorcycle m_i crashed into the trace s_j, see Figure 52. We may assume without loss of generality that s_i is right of $\overrightarrow{s_j}$. First we note that by our choice of λ we may apply Lemma 3.4. We denote by p the intersection $s_i \cap s_j$. Further, we set $\epsilon := \mu/8$, which allows us to apply Lemma 3.6, because

$$\max\left\{2, \frac{8L}{\mu}\right\} \geq \max\left\{2, \frac{L}{\min\{\mu/4, \epsilon\}}\right\}.$$

Thus, the endpoint q of the reflex arc that is traced out by u_i, has a distance of at most $\mu/8$ to p. On the other hand, u_j reaches p at time $L/\lambda|v_j|$ at the latest. We conclude that the right arm of m_j reaches q at time $L/\lambda|v_j| + \mu/8$ at the latest, which is bounded from above by

$$\frac{L \cdot \mu \cdot \min_k |v_k| \cdot \sin\Phi}{8 \cdot L \cdot |v_j|} + \frac{\mu}{8} \leq \frac{\mu}{4}$$

by our choice of λ. Summarizing, the point q is swept by the wavefront of the right arm of m_j and is reached by u_i until $\mu/4$ time. Hence, u_i must have caused a split event until the requested time by crashing into the wavefront of the right arm of m_j.

For the second case assume that m_i escapes. Lemma 3.4 and Lemma 3.5 imply that u_i does not lead to a split event until time $\mu/4$. □

The previous lemma enables us to compute the motorcycle graph by employing a straight skeleton algorithm. However, in order to apply the lemma, we need to compute appropriate values for L, Φ, μ in order to determine a sufficiently large λ. While L and Φ are already given independent of $\mathcal{M}(m_1, \ldots, m_n)$, the following lemma gives a formula for μ, for which the actual motorcycle graph is not needed to be known. (In the following lemma we take $d(\overrightarrow{s_i}, \emptyset)$ to be infinity.)

Lemma 3.9. *For any two disjoint motorcycle traces s_i and s_j the Minkowski sums $s_i + D_\mu$ and $s_j + D_\mu$ are disjoint for*

$$\mu := \frac{1}{3} \min_{1 \leq i,j,k \leq n} \mathbb{R}^+ \cap \{d(\overrightarrow{s_i}, p_j), d(\overrightarrow{s_i}, \overrightarrow{s_j} \cap \overrightarrow{s_k})\}.$$

Proof. In order to guarantee that the Minkowski sums are disjunct it suffices to show that μ is a lower bound of a third of the minimum of all pairwise infimum distances of disjunct traces s_i and s_j.

Let us consider two disjunct traces s_i and s_j. We choose two points $q_i \in s_i, q_j \in s_j$ for which $d(s_i, s_j) = d(q_i, q_j)$ holds. We may assume that either q_i is an endpoint of s_i or q_j is an endpoint of s_j, because the infimum distance is not uniquely attained for two interior points of s_i and s_j. (If s_k is not bounded, the only endpoint is p_k.) If q_j is the start point of s_j then we have $d(s_i, s_j) = d(s_i, p_j) \geq 3\mu$. If q_j is the opposite endpoint of s_j — and hence s_j is a segment — then s_j crashed into some other motorcycle trace. Hence, there is a trace s_k such that $q_j = s_j \cap s_k$. Again we get $d(s_i, s_j) = d(s_i, q_j) \geq 3\mu$. Analogous arguments hold if q_i is an endpoint of s_i. □

After computing appropriate values for L, Φ and μ for a set of motorcycles $m_1, \ldots, m_n$, we can determine a sufficiently large λ and build the input graph G by constructing the triangles $\Delta_1, \ldots, \Delta_n$ as described. After computing the straight skeleton $\mathcal{S}(G)$, we determine the length of each trace by applying the conditions listed in Lemma 3.8.

3.4.3 Constructing the straight skeleton is P-complete

The concept of P-completeness is similar to the concept of NP-completeness. A problem A from P is said to be P-complete under NC-reductions if any problem in P can be reduced to A in NC time. The complexity class NC comprises all problems that can be efficiently solved in parallel. That is, they can be solved in poly-logarithmic time complexity using a polynomial number of processors. Hence, all P-complete problems cannot be efficiently solved using parallel computers, unless NC = P. In other words, P-complete problems are inherently sequential, provided that NC ≠ P. If the latter condition would be wrong then every problem solvable in polynomial time could also be solved efficiently in parallel. To show this it would suffice to find a single P-complete problem that can be solved in NC time. However, is is commonly assumed that NC ≠ P.

In order to show that a specific problem A is P-complete it suffices to reduce any P-complete problem to A in NC time. Alternatively, one could also seek for an according LOGSPACE reduction, because LOGSPACE ⊂ NC. For further details on P-completeness we refer to the book by Greenlaw, Hoover and Ruzzo [GHR95].

Atallah et al. [ACG93] described a framework for geometric reductions of the P-complete Planar Circuit Value problem and used it to prove the P-completeness of several geometric problems. For their framework, they consider specific binary circuits which are organized in layers: an input layer at the top and alternating routing layers and logic layers from top to bottom. The routing layers and logic layers consist of rows of components occupying non-overlapping rectangles. A logic layer consists of a row of $\vee$-gates and a row of $\neg$-gates and a routing layer comprises rows of routing components, namely left shifts, right shifts, fan-out gates and vertical wires. The binary circuit has a single output gate and the Circuit Value problem asks for the output value of this gate when a specific input vector is presented to the input gates. Atallah et al. proved that the Circuit Value problem for circuits of such a specific layout is still P-complete.

Investigating the P-completeness of geometric problems often requires the availability of exact geometric computations, which are not in NC. For instance, they mention the problem of determining whether four points lie on a circle, which is an essential predicate when computing Voronoi diagrams. In order to investigate the P-completeness of geometric problems,

Attalah et al. [ACG93] propose that the answers to basic geometric queries are provided by an oracle.

A basic building block for showing that the straight skeleton is P-complete is the construction of the triangles Δ_i. Assume p_i, α_i, v_i, μ, and λ are given. We further assume that an oracle determines the intersection points of two circles with given centers and radii. Then, we can construct Δ_i as follows. We first compute the point $q_i = p_i + \lambda v_i$, which is the position of m_i at time one, see Figure 48. Then we construct the circle C_1 with $[p_i, q_i]$ as diameter and the circle C_2 centered at p_i with radius 1. The two circles C_1, C_2 intersect at two points, say a_i, b_i. The triangle Δ_i^* with vertices a_i, b_i, q_i is an isosceles triangle with angle $2\alpha_i$ at q_i and therefore similar to Δ_i. (Note that p_i, a_i, q_i form a right-angled triangle within Thales' circle C_1.) The length of the arms of Δ_i^* at q_i are at most $\lambda|v_i|$. By scaling the triangle by the factor $\mu/2\lambda|v_i|$ and by translating it accordingly, we get a triangle with the desired geometry. (Strictly speaking, the arms of the constructed triangle are a bit shorter than $\mu/2$, but this is only to our advantage.)

Eppstein and Erickson [EE99] proved that the computation of the motorcycle graph is P-complete by presenting a LOGSPACE-reduction of the Circuit Value problem to the computation of the motorcycle graph. Eppstein and Erickson demonstrated how to translate the Circuit Value problem to the motorcycle graph construction problem by simulating each gadget of the binary circuit using motorcycles. The values 1 and 0 on a wire are represented by the presence or absence of a motorcycle on a track. The original question for the output value of a particular gate of the circuit can be translated to the question whether a specific motorcycle crashes until some distance from its start point. In other words, Eppstein and Erickson proved that the decision problem whether a specific motorcycle crashes until some distance from its start point is P-complete.

Lemma 3.10. *The construction of the straight skeleton of a planar straight-line graph is P-complete under LOGSPACE-reductions.*

Proof. Eppstein and Erickson reduced the Circuit Value problem to a specific motorcycle graph problem. The next step is to reduce the motorcycle graph problem to the straight-skeleton problem: we construct a suitable input graph G that allows us to apply Lemma 3.8 in order to decide whether a specific motorcycle crashes until some distance from its start point.

According to [EE99] all $O(1)$ different types of motorcycle gadgets are arranged in an $n \times n$ grid. Each gadget takes constant space and consists of $O(1)$ motorcycles. To determine a sufficiently large λ, we need bounds on L, Φ and μ. An upper bound on L is the length of the diagonal of the $n \times n$ grid. Further, $\sin\Phi \geq 1/2$ since the direction angles of the motorcycles are all multiples of $\pi/4$. A lower bound on μ can be obtained by applying Lemma 3.9 on each gadget independently and taking the minimum among them. Finally, we build G by modeling each motorcycle (independently from each other) as an isosceles triangle, as described in Section 3.4.1. □

We can easily extend the construction of G to form a polygon with holes, by adding a sufficiently large bounding box to G. As remarked in [EE99], only one motorcycle m may leave the bounding box B of the $n \times n$ grid. The motorcycle m encodes the output of the

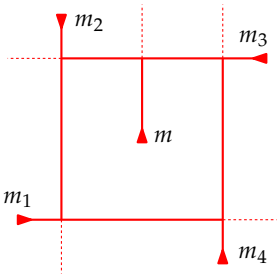


Figure 53: These motorcycle traces cannot be approximated easily by a straight skeleton of a simple polygon.

binary circuit by leaving B if the circuit evaluates to 1 and by crashing within B if the circuit evaluates to 0.

By Lemma 3.8 the reflex wavefront vertex v, which corresponds to m, encodes the output of the binary circuit by leading to a split event until time $\mu/4$ if and only if the circuit evaluates to 0. Lemma 3.4 implies that the wavefront vertices stay within $B + D_\mu$, except possibly v. Hence, we could enlarge B by 2μ at each side and add it to G such that the wavefronts of B do not interfere with the wavefronts of the triangles until time $\mu/4$, except for v. Still, we can determine the output of the binary circuit by checking whether the reflex straight-skeleton arc that corresponds to v, ends within $B + D_\mu$ until time $\mu/4$. (Recall that the end of a reflex straight-skeleton arc marks the place where the reflex wavefront vertex led to a split event.)

Corollary 3.11. *The construction of the straight skeleton of a polygon with holes is P-complete under LOGSPACE-reductions.*

Unfortunately, our P-completeness proof cannot be applied easily to simple polygons. Consider the five motorcycles depicted in Figure 53. A simple polygon, whose straight skeleton would approximate the motorcycle traces, would need to connect the start points of m and $m_1, \ldots, m_4$. But in order to decide where the red square, formed by the traces of $m_1, \ldots, m_4$, can be penetrated by the polygon, while avoiding to stop a motorcycle too early, it would be necessary to know that a specific motorcycle m_i crashes into a specific motorcycle m_j. However, deciding whether a specific motorcycle crashes does not seem much easier than computing the whole motorcycle graph. Hence, it remains open whether the computation of straight skeletons of polygons is P-complete.

4 CONCLUDING REMARKS

The investigations done in this thesis are driven by the lack of an efficient implementation of straight skeletons for real-world purposes that stands opposite to the large number of different industrial and academical applications. At the moment, the state-of-the-art straight-skeleton implementation is shipped with the CGAL library. However, our experiments illustrate that the CGAL implementation is only applicable to real-world problems in a limited manner: Both, the runtime and the memory footprint of the CGAL implementation increase at least quadratically with the input size. While time is more or less an infinite resource, the opposite is true for space. The quadratic memory consumption is an issue that basically renders the implementation in CGAL inapplicable to datasets containing more than ten thousand vertices.

We started our investigations towards an efficient straight-skeleton algorithm with an analysis of the triangulation-based approach by Aichholzer and Aurenhammer [AA98] in Section 2.2. The gap between $\Omega(n^2)$ and $O(n^3)$ for the worst-case number of flip events leads to an open question regarding the time complexity of the algorithm. We present different results regarding this gap and finally prove the existence of Steiner triangulations that are free of flip-events. This result is primarily of theoretical interest, since we use the straight skeleton for the placement of the Steiner points. However, the fact that flip-event-free Steiner triangulations exist led to a novel wavefront-type straight-skeleton algorithm for simple non-degenerate polygons which is based on the motorcycle graph in Section 2.3.

In order to make this algorithm applicable to real-world input we needed to remove the non-degeneracy assumption on the input. In Section 2.4, we carefully generalized the motorcycle graph in order to reflect so-called vertex events for straight skeletons. That is, the simultaneous crash of two or more motorcycles into each other may require to start an additional motorcycle. We demand two essential geometric properties for the generalized motorcycle graph: (i) $\mathcal{M}(G)$ needs to cover the reflex arcs of $\mathcal{S}(G)$, also in the presence of vertex events, and (ii) $G + \mathcal{M}(G)$ needs to induce a convex tessellation of the plane. These two properties allowed us to extend our algorithm to arbitrary planar straight-line graphs G in Section 2.5. Furthermore, the generalization of the motorcycle graph leads to an alternative characterization of the straight skeleton of arbitrary planar straight-line graphs by extending the characterization of Cheng and Vigneron and it motivated a straight-skeleton algorithm that is based on 3D graphics hardware.

The wavefront-type straight-skeleton algorithm presented in Section 2.5 is easy to implement, has a worst-case runtime of $O(n^2 \log n)$ and operates in $O(n)$ space. The resulting implementation Bone uses ordinary double-precision floating-point arithmetic and accepts planar straight-line graphs as input. Our experiments showed an $O(n \log n)$ runtime on 13 500 datasets of different types. This constitutes an improvement of a linear factor in time and space compared to the implementation in CGAL, which only accepts polygons with

holes as input. On datasets with several thousand vertices, our implementation is up to two orders of magnitude faster and we need less than 100 megabytes of memory instead of several gigabytes. Moreover, due to our low memory requirements, we are able to compute the straight skeleton of datasets with a million vertices. These circumstances make Bone the fastest straight-skeleton implementation at the moment and the first extensively tested implementation that is capable to process planar straight-line graphs originating from real-world applications.

In order to push Bone to industrial strength, we plan to enhance the numerical stability in the presence of parallel wavefronts that collapse simultaneously. Detecting the simultaneous collapse of larger parts of the wavefront is simplified by exploiting the convex tessellation induced by the motorcycle graph. From an abstract point of view we gain from the fact that the motorcycle graph already provides a sufficient amount of information on the topology of the straight skeleton. Bone is currently extended to use different numerical backends by colleagues, including the arbitrary precision library MPFR [MPF] and the exact geometric computation library CORE [Cor]. The necessary work for MPFR is almost finished and preliminary runtime tests with a floating-point precisions of 212 and 1000 bits show a drop in performance by a factor of approximately 10–20, which is still reasonable for real-world applications.

Even though Bone exhibits an $O(n \log n)$ runtime in practice, the gap between the lower bound of $\Omega(n \log n)$ and the theoretically fastest algorithm by Eppstein and Erickson [EE99], with a theoretical worst-case time complexity of $O(n^{17/11+\epsilon})$, remains open. If the motorcycle graph $\mathcal{M}(G)$ is known, the algorithm behind Bone has an $O((n+k)\log n)$ time complexity, where $k \in O(nr)$ denotes the number of switch events and $r \in O(n)$ denotes the number of reflex wavefront vertices. In order to obtain a worst-case runtime of $O(n \log n)$ it would be necessary to bound the number k of switch events to $O(n)$. One approach towards this goal could be the introduction of additional motorcycles that are launched from convex wavefront vertices of $\mathcal{W}(G, 0)$ such that the number of interactions between moving Steiner vertices and convex wavefront vertices is reduced to $O(n)$. However, in order to extend the basic algorithm behind Bone accordingly, we would require that any extension of the motorcycle graph $\mathcal{M}(G)$ by additional motorcycles needs to maintain the validity of Lemma 2.25 and Theorem 2.26: $G + \mathcal{M}(G)$ needs to induce a convex tessellation and the motorcycle traces need to cover the reflex straight-skeleton arcs.

We also look forward to generalize the algorithms behind Bone in order to compute weighted straight skeletons. In principal, wavefront-type algorithms tend to have a straightforward generalization from the unweighted straight skeleton to the weighted counterpart. However, in order to generalize Bone to weighted straight skeletons, it is necessary to adapt the definition of the motorcycle graph such that Lemma 2.25 and Theorem 2.26 still hold in the weighted case. First of all, the speed and direction of each motorcycle is given by Lemma 1.11 in order to match the speeds of the corresponding reflex wavefront vertices. Assume for a moment that we simply transfer the rules for launching new motorcycles from Section 2.4.1. If we do so, Lemma 2.25 remains true for the Case (d) in Figure 31, but not for Case (c). Note that if a vertex event happens then the reflex straight-skeleton arcs do not need to tessellate a local disk into convex slices. That is, the lower chains of the faces do not need to be convex. In fact, the faces do not even need to be monotone. In order to fix Lemma 2.25, we could launch a second motorcycle that continues the movement of the right

ancestor, as in Case (d). However, the question remains whether the number of motorcycles is still in $O(n)$ after this adaption. In the next step, we need to guarantee that Theorem 2.26 remains true for this generalized motorcycle graph. Note that again the tilted motorcycle traces $\hat{m}_1, \ldots, \hat{m}_k$ with the same right arm e lie on the supporting plane of the tilted face $\hat{f}(e)$ and at least the Step (i) in the proof of Theorem 2.26 remains true. We also want to note that a generalization of Theorem 2.26 could also provide a lower envelope characterization of the weighted straight skeleton, which does not exist so far. Recall that Eppstein and Erickson [EE99] showed that the trivial generalization does not work, see Figure 16.

In Chapter 3 we took a closer look at the motorcycle graph. In order to compute straight skeletons fast in practice by means of Bone, we require an implementation for the generalized motorcycle graph that performs well on real-world data. We started with a stochastic analysis of the average trace length in a motorcycle graph in Section 3.2. It turned out that if the motorcycles are distributed uniformly within the unit square then we can expect an average trace length of $\Theta(1/\sqrt{n})$. In other words, if we impose a regular $\sqrt{n} \times \sqrt{n}$ grid on the unit square then we expect that a motorcycle crosses $O(1)$ grid cells on average. This fact motivated a simple pragmatic approach: we enhanced the straight-forward approach based on a priority-queue with geometric hashing in Section 3.3. This measurement reduces the runtime from $O(n^2 \log n)$ to $O(n \log n)$ for input data where the motorcycles are distributed uniformly enough. We performed extensive runtime tests with our implementation Moca. For the vast majority of our datasets Moca runs in $O(n \log n)$ time, which is one of the key reasons for the good overall performance of our straight-skeleton code Bone. Finally, we also used Moca in order to substantiate the theoretical predictions provided by the stochastic analysis.

We plan to further boost the performance of Moca by implementing the $O(n \log n)$ algorithm to compute the motorcycle graph outside the convex hull of the start points, see Section 3.3.4. Furthermore, in order to make the runtime performance of Moca more robust against clustered start points, one may also replace the ordinary rectangular grid by quad trees, which allows a non-uniform tessellation of the plane. Additionally, quad trees would enable us to dynamically increase the tessellation depth at regions where motorcycles accumulate during the propagation process.

Our final contribution concerns the geometric relation of motorcycle graphs and straight skeletons. By Theorem 2.26 we know that the reflex straight-skeleton arcs of $\mathcal{S}(G)$ approximate the motorcycle traces of $\mathcal{M}(G)$ up to a specific extent. Furthermore, one observers that in general the gap between the reflex arcs and the motorcycle traces decreases if the speed of the motorcycle increases. In Section 3.4, we first show that we can construct a planar straight-line graph G whose straight skeleton approximates the motorcycle graph. Based on this result, we present a simple algorithm that computes the motorcycle graph using the straight skeleton. Finally, we show that the resulting algorithm admits a LOGSPACE-reduction of the motorcycle graph problem to the straight-skeleton problem. Consequently, the P-completeness of the motorcycle graph by Eppstein and Erickson [EE99] implies the P-completeness of the straight-skeleton problem for planar straight-line graphs and for polygons with holes. We want to note that Eppstein and Erickson [EE99] were the first to mention that straight skeleton is P-complete, but no proof was given. The P-completeness of straight skeletons has important practical implications: no efficient parallel algorithms exist to compute straight skeletons of planar straight-line graphs and polygons with holes,

provided that $\mathsf{P} \neq \mathsf{NC}$. We also note that it is desirable to find an efficient sequential reduction of motorcycle graphs to straight skeletons in order to transfer lower bounds from motorcycle graphs to straight skeletons in the sequential manner.

A NOTATION

$A + B$	The Minkowski sum $A + B = \{x + y \ : \ x \in A, y \in B\}$ for two point sets A and B.
$\overline{pq}, \overline{e}, \overline{\hat{f}(e)}$	The supporting line of the points p, q resp. the edge e and the supporting plane of the lifted face $\hat{f}(e)$.
$[pq]$	The straight-line segment between two points p and q.
$\hat{a}, \hat{s}, \hat{m}, \hat{f}(e)$	The lifted counterpart of the arc a, motorcycle trace s, motorcycle m, straight-skeleton face $f(e)$ in the terrain model.
$d(p, q)$	The Euclidean distance between two points p and q.
$d(A, B)$	The infimum distance $\inf_{x \in A, y \in B} d(x, y)$ for two point sets A and B.
D_r	A disk with radius r and the origin as center.
$e(t)$	The union of the straight-line segments occupied by the wavefront edge e at time t, see Definition 2.2.
$\overline{e(t)}$	The supporting line of $e(t)$. If e is emanated by a terminal vertex or an isolated vertex v of G then we define $\overline{e(0)} := \lim_{t \searrow 0} \overline{e(t)}$, see Definition 2.2.
$f(e)$	The straight-skeleton face of the wavefront edge e, see Definition 1.1.
G	A planar straight-line graph (with no isolated vertices).
$\mathcal{M}(m_1, \ldots, m_n)$	The motorcycle graph of the motorcycles $m_1, \ldots, m_n$, see Definition 1.8.
$\mathcal{M}(P)$	The motorcycle graph induced by the simple polygon P, see Definition 1.9.
$\mathcal{M}(G)$	The motorcycle graph induced by a planar straight-line graph G, see Definition 2.23.
P	A simple polygon in the plane (with holes if mentioned explicitly).
$\mathcal{S}(P)$	The straight skeleton of a simple polygon P, only considered within the polygon P, see Definition 1.1.
$\mathcal{S}(G)$	The straight skeleton of a planar straight-line graph G, see Section 1.2.2.
$\mathcal{T}(G)$	The terrain model corresponding to $\mathcal{S}(G)$, see Definition 1.5.
$\mathcal{W}(G, t)$	The wavefront of G at time $t \geq 0$, see Definition 1.4.

B EXAMPLES

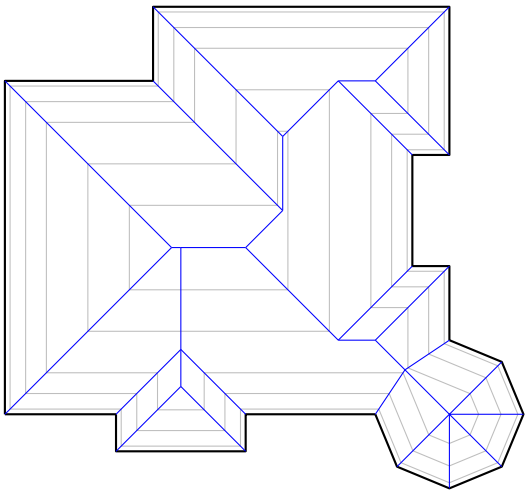


Figure 54: The straight skeleton and offset curves of the polygon that forms the walls of the house that is shown in Figure 8.

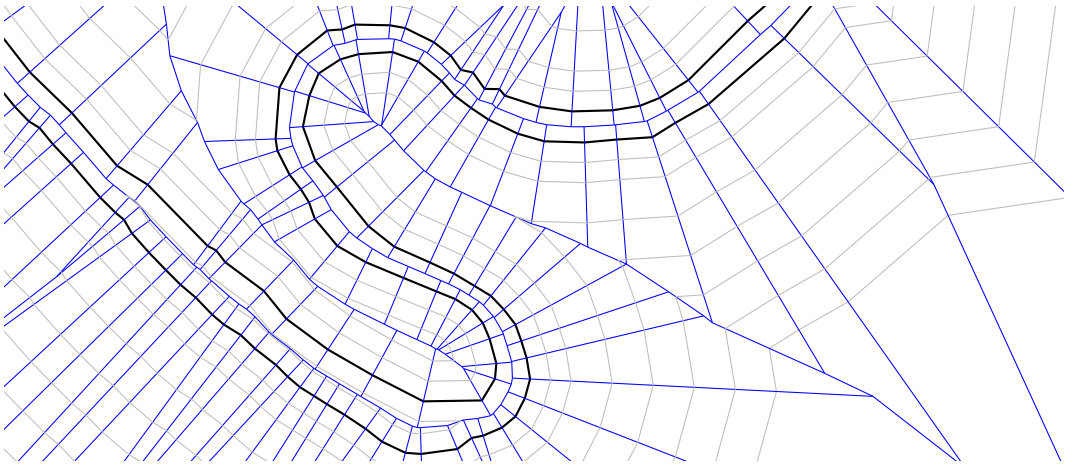


Figure 55: The straight skeleton and offset curves of the river bank of the Danube in Figure 9.

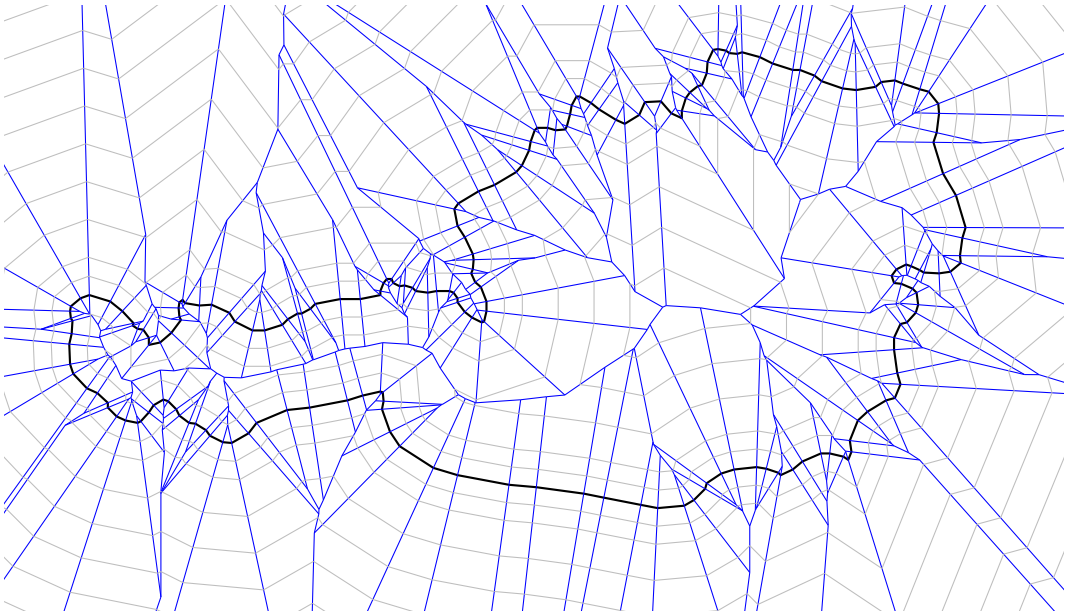


Figure 56: The straight skeleton and offset curves of the outline of Austria.

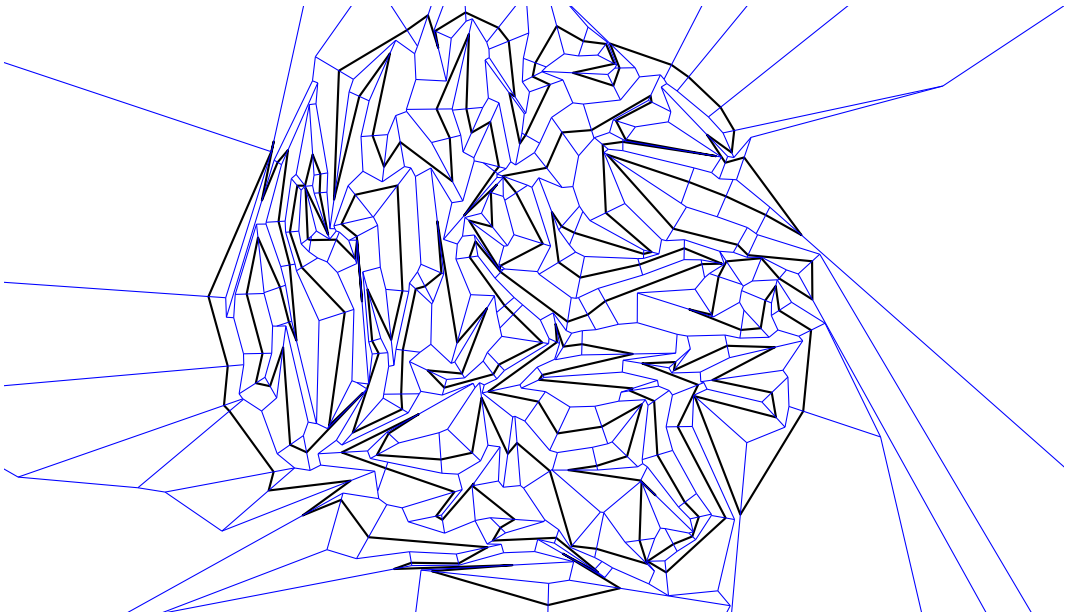


Figure 57: The straight skeleton of a random polygon generated by RPG.

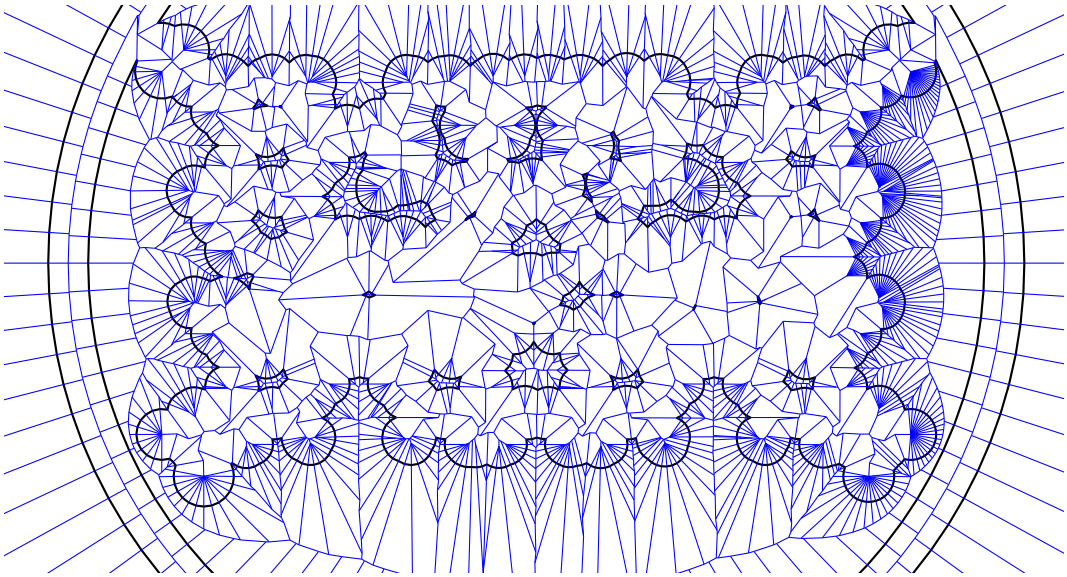


Figure 58: The straight skeleton of a polygon with holes.

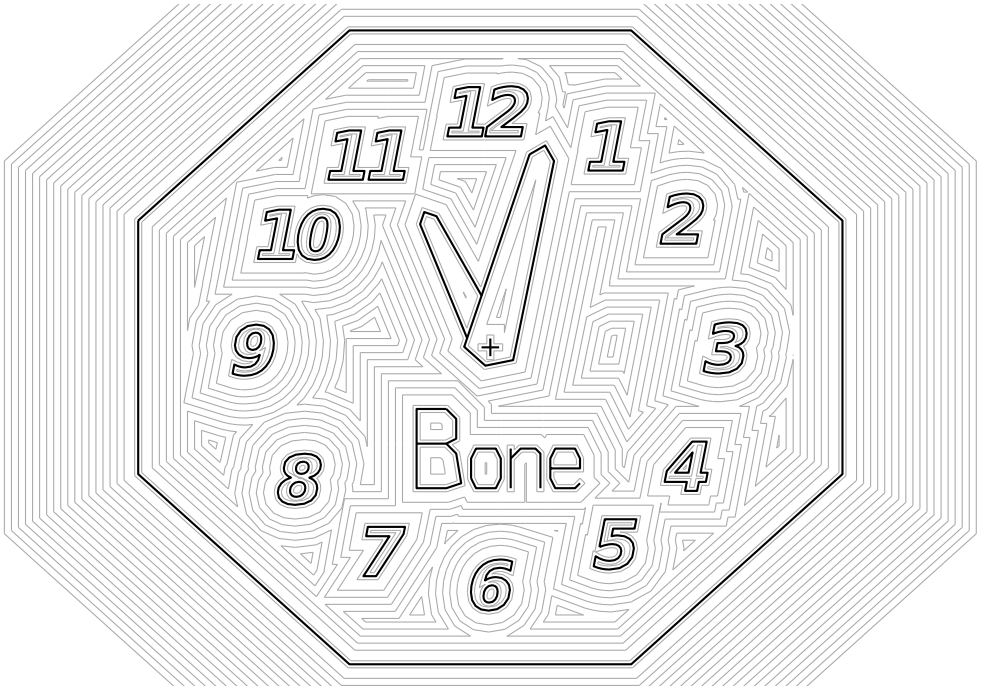


Figure 59: Offset curves based on the straight skeleton of a planar straight-line graph.

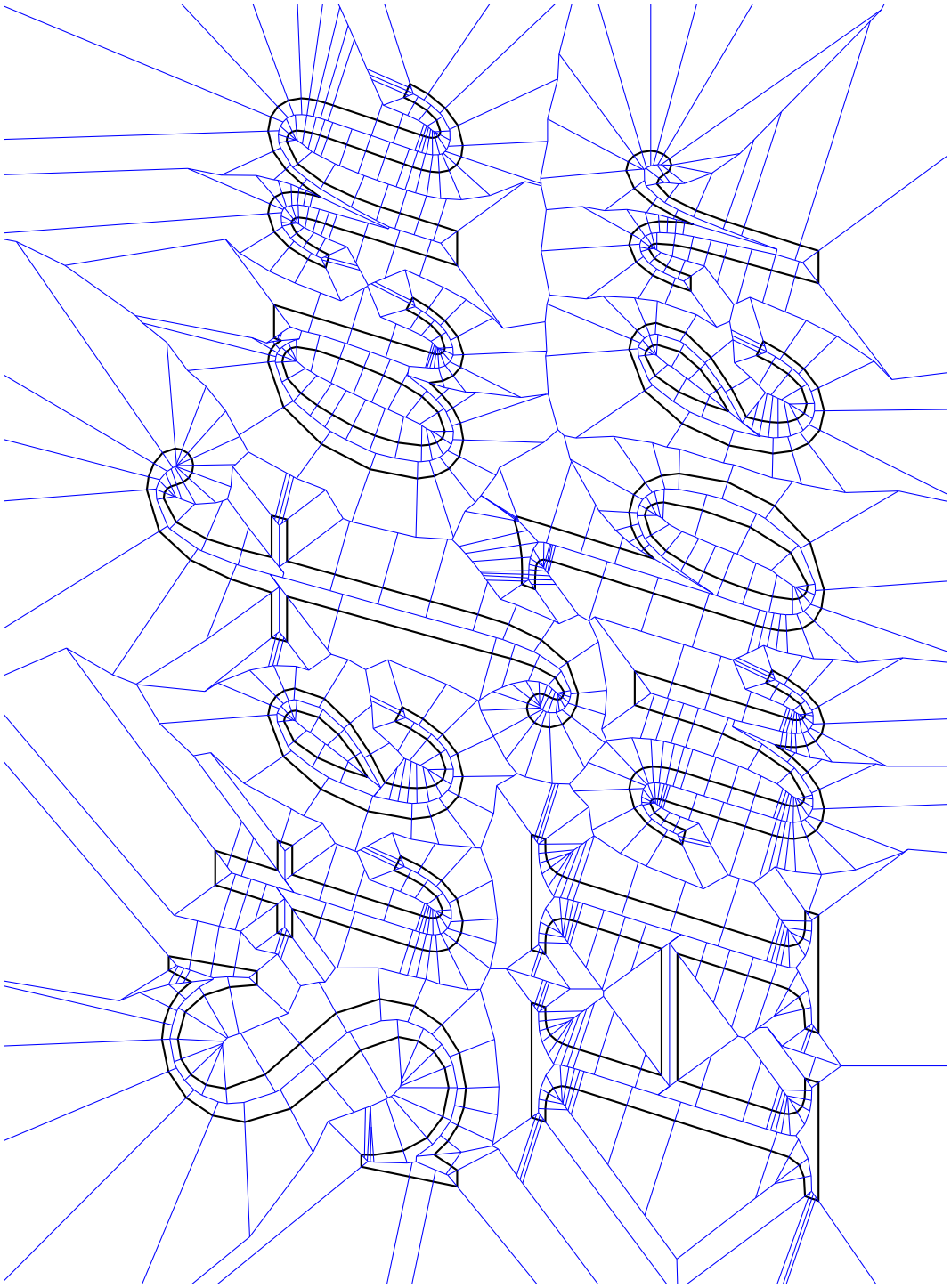


Figure 60: The straight skeleton of a font outline.

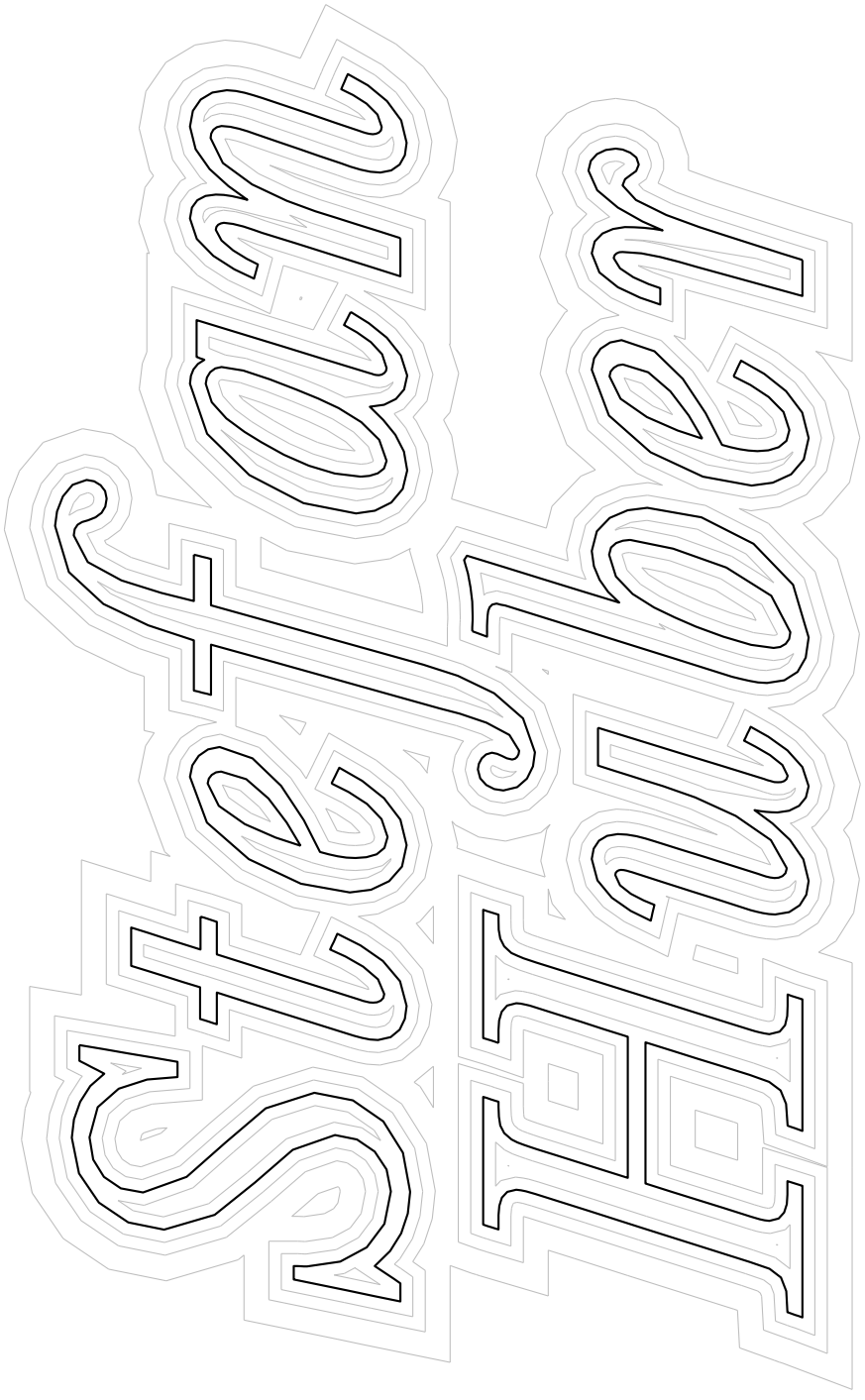


Figure 61: Offset curves of a font outline.

BIBLIOGRAPHY

[AA96] O. Aichholzer and F. Aurenhammer. Straight Skeletons for General Polygonal Figures. In *Proc. 2nd Annu. Internat. Conf. Comput. Combinatorics*, volume 1090 of *Lecture Notes Comput. Sci.*, pages 117–126. Springer-Verlag, 1996.

[AA98] O. Aichholzer and F. Aurenhammer. Straight Skeletons for General Polygonal Figures in the Plane. In A.M. Samoilenko, editor, *Voronoi's Impact on Modern Science, Book 2*, pages 7–21. Institute of Mathematics of the National Academy of Sciences of Ukraine, Kiev, Ukraine, 1998.

[AAAG95] O. Aichholzer, D. Alberts, F. Aurenhammer, and B. Gärtner. Straight Skeletons of Simple Polygons. In *Proc. 4th Internat. Symp. of LIESMARS*, pages 114–124, Wuhan, P.R. China, 1995.

[AAP04] O. Aichholzer, F. Aurenhammer, and B. Palop. Quickest Paths, Straight Skeletons, and the City Voronoi Diagram. *Discrete Comput. Geom.*, 31(1):17–35, 2004.

[ACG93] M.J. Atallah, P.B. Callahan, and M.T. Goodrich. P-complete Geometric Problems. *Internat. J. Comput. Geom. Appl.*, 3(4):443–462, 1993.

[AE99] P. K. Agarwal and J. Erickson. Geometric Range Searching and Its Relatives. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, volume 223 of *Contemporary Mathematics*, pages 1–56. American Mathematical Society, 1999.

[AGSS87] A. Aggarwal, L.J. Guibas, J. Saxe, and P. Shor. A Linear Time Algorithm for Computing the Voronoi Diagram of a Convex Polygon. In *Proc. 19th Annu. ACM Sympos. Theory Comput.*, pages 39–45, 1987.

[AH96] T. Auer and M. Held. Heuristics for the Generation of Random Polygons. In *Proc. Canad. Conf. Comput. Geom. (CCCG'96)*, pages 38–44, Ottawa, Canada, Aug 1996. Carleton University Press.

[AM94] P. Agarwal and J. Matoušek. On Range Searching with Semialgebraic Sets. *Discrete Comput. Geom.*, 11:393–418, 1994.

[AS95] H. Alt and O. Schwarzkopf. The Voronoi Diagram of Curved Objects. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 89–97, Vancouver, BC, Canada, 1995.

[BEGV08] G. Barequet, D. Eppstein, M. T. Goodrich, and A. Vaxman. Straight Skeletons of Three-Dimensional Polyhedra. In *Proc. 16th Annu. Europ. Symp. Algorithms (ESA '08)*, pages 148–160, Karlsruhe, Germany, Sep 2008.

[BGLSS04] G. Barequet, M.T. Goodrich, A. Levi-Steiner, and D. Steiner. Contour Interpolation by Straight Skeletons. *Graph. Models*, 66(4):245–260, 2004.

[Ble] Blender. `http://www.blender.org/`. last checked on May, 2011.

[BO79] J. Bentley and T. Ottmann. Algorithms for Reporting and Counting Geometric Intersections. *IEEE Trans. Comput.*, C-28:643–647, 1979.

[Cac04] F. Cacciola. A CGAL Implementation of the Straight Skeleton of a Simple 2D Polygon with Holes. In *2nd CGAL User Workshop*, Polytechnic Univ., Brooklyn, New York, USA, June 2004.

[CEG+91] B. Chazelle, H. Edelsbrunner, M. Grigni, L. Guibas, J. Hershberger, M. Sharir, and J. Snoeyink. Ray Shooting in Polygons Using Geodesic Triangulations. In *Proc. 18th Internat. Colloq. Automata Lang. Program.*, number 510 in Lecture Notes Comput. Sci., pages 661–673. Springer-Verlag, 1991.

[CGA] CGAL — Computational Geometry Algorithms Library. `http://www.cgal.org`. accessed on May, 2011.

[Cha91] B. Chazelle. Triangulating a Simple Polygon in Linear Time. *Discrete Comput. Geom.*, 6:485–524, 1991.

[Cha93] B. Chazelle. Cutting Hyperplanes for Divide-and-Conquer. *Discrete Comput. Geom.*, 9:145–158, Apr 1993.

[Cha04] B. Chazelle. Cuttings. In *Handbook of Data Structures and Applications*, pages 25.1–25.10. 1 edition, 2004. Chapman and Hall/CRC Press.

[Cor] CORE library project. `http://cs.nyu.edu/exact/core_pages/`. accessed on May, 2011.

[CRU89] J. Czyzowicz, I. Rival, and J. Urrutia. Galleries, Light Matchings and Visibility Graphs. In *Proc. 1st Workshop Algorithms Data Struct.*, pages 316–324, Ottawa, Canada, Aug 1989.

[CSW99] F. Chin, J. Snoeyink, and C.A. Wang. Finding the Medial Axis of a Simple Polygon in Linear Time. *Discrete Comput. Geom.*, 21(3):405–420, 1999.

[CV02] S.-W. Cheng and A. Vigneron. Motorcycle Graphs and Straight Skeletons. In *Proc. 13th ACM-SIAM Sympos. Discrete Algorithms*, pages 156–165, San Francisco, CA, USA, 2002.

[CV07] S.-W. Cheng and A. Vigneron. Motorcycle Graphs and Straight Skeletons. *Algorithmica*, 47:159–182, Feb 2007.

[DDL98] E. D. Demaine, M. L. Demaine, and A. Lubiw. Folding and Cutting Paper. In *Revised Papers from the Japan Conference on Discrete and Computational Geometry (JCDCG'98)*, volume 1763 of *Lecture Notes Comput. Sci.*, pages 104–117, Tokyo, Japan, Dec 1998.

[DDLS05] E. D. Demaine, M. L. Demaine, J. F. Lindy, and D. L. Souvaine. Hinged Dissection of Polypolyhedra. In *Proc. 9th Workshop Algorithms Data Struct. (WADS 2005)*, volume 3608 of *Lecture Notes Comput. Sci.*, pages 205–217, Waterloo, Ontario, Canada, Aug 2005.

[DDM00] E.D. Demaine, M.L. Demaine, and J.S.B. Mitchell. Folding Flat Silhouettes and Wrapping Polyhedral Packages: New Results in Computational Origami. *Comput. Geom. Theory and Appl.*, 16(1):3–21, May 2000.

[DMN+10] G. K. Das, A. Mukhopadhyay, S. C. Nandy, S. Patil, and S. V. Rao. Computing the Straight Skeleton of a Monotone Polygon in $O(n \log n)$ Time. In *Proc. 22nd Canad. Conf. Comput. Geom. (CCCG 2010)*, pages 207–210, Winnipeg, Canada, Aug 2010.

[DO07] E. D. Demaine and J. O'Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press, Jul 2007.

[EE99] D. Eppstein and J. Erickson. Raising Roofs, Crashing Cycles, and Playing Pool: Applications of a Data Structure for Finding Pairwise Interactions. *Discrete Comput. Geom.*, 22(4):569–592, 1999.

[EGKT08] D. Eppstein, M.T. Goodrich, E. Kim, and R. Tamstorf. Motorcycle Graphs: Canonical Quad Mesh Partitioning. *Comput. Graph. Forum*, 27(5):1477–1486, Sep 2008.

[Epp00] D. Eppstein. Fast hierarchical clustering and other applications of dynamic closest pairs. *J. Exp. Algorithmics*, 5, Dec 2000.

[FO99] P. Felkel and Š. Obdržálek. Improvement of Oliva's Algorithm for Surface Reconstruction from Contours. In *Proc. 15th Spring Conf. Comput. Graphics*, pages 254–263, Budmerice, Slovakia, Apr 1999.

[For00] Steven Fortune. Introduction. *Algorithmica*, 27(1):1–4, 2000.

[GHR95] R. Greenlaw, H. J. Hoover, and W. L. Ruzzo. *Limits to Parallel Computation: P-Completeness Theory*. Oxford University Press, Apr 1995.

[GLI] GNU C Library. `http://www.gnu.org/software/libc`. accessed on May, 2011.

[Hav05] S. Havemann. *Generative Mesh Modeling*. PhD thesis, TU Braunschweig, Braunschweig, Germany, 2005.

[HCK+99] K. Hoff, T. Culver, J. Keyser, M. Lin, and D. Manocha. Fast Computation of Generalized Voronoi Diagrams Using Graphics Hardware. In *Comput. Graphics (SIGGRAPH '99 Proc.)*, pages 277–286, Los Angeles, CA, Aug 1999.

[Hel91] M. Held. *On the Computational Geometry of Pocket Machining*, volume 500 of *Lecture Notes Comput. Sci.* Springer-Verlag, June 1991. ISBN 3-540-54103-9.

[Hel94] M. Held. On Computing Voronoi Diagrams of Convex Polyhedra by Means of Wavefront Propagation. In *Proc. 6th Canad. Conf. Comput. Geom. (CCCG'94)*, pages 128–133, Saskatoon, Saskatchewan, Canada, Aug 1994.

[Hel01] M. Held. VRONI: An Engineering Approach to the Reliable and Efficient Computation of Voronoi Diagrams of Points and Line Segments. *Comput. Geom. Theory and Appl.*, 18(2):95–123, Mar 2001.

[HH09a] M. Held and S. Huber. Topology-Oriented Incremental Computation of Voronoi Diagrams of Circular Arcs and Straight-Line Segments. *Comput. Aided Design*, 41(5):327–338, May 2009.

[HH09b] S. Huber and M. Held. A Practice-Minded Approach to Computing Motorcycle Graphs. In *Proc. 25th Europ. Workshop Comput. Geom.*, pages 305–308, Brussels, Belgium, Mar 2009.

[HH10a] S. Huber and M. Held. Computing Straight Skeletons of Planar Straight-Line Graphs Based on Motorcycle Graphs. In *Proc. 22nd Canad. Conf. Comput. Geom. (CCCG 2010)*, pages 187–190, Winnipeg, Canada, Aug 2010.

[HH10b] S. Huber and M. Held. Straight Skeletons and their Relation to Triangulations. In *Proc. 26th Europ. Workshop Comput. Geom.*, pages 189–192, Dortmund, Germany, Mar 2010.

[HH11a] S. Huber and M. Held. Approximating a Motorcycle Graph by a Straight Skeleton. 2011. (submitted for publication).

[HH11b] S. Huber and M. Held. Motorcycle Graphs: Stochastic Properties Motivate an Efficient Yet Simple Implementation. *J. Exp. Algorithmics*, 2011. (in press).

[HH11c] S. Huber and M. Held. Theoretical and Practical Results on Straight Skeletons of Planar Straight-Line Graphs. In *Proc. 27th Annu. ACM Sympos. Comput. Geom.*, Paris, France, to be published 2011.

[HLA94] M. Held, G. Lukács, and L. Andor. Pocket Machining Based on Contour-Parallel Tool Paths Generated by Means of Proximity Maps. *Comput. Aided Design*, 26(3):189–203, Mar 1994.

[HP00] S. Har-Peled. Constructing Planar Cuttings in Theory and Practice. *SIAM J. Comput.*, 29(6):2016–2039, 2000.

[HS08] J.-H. Haunert and M. Sester. Area Collapse and Road Centerlines based on Straight Skeletons. *GeoInformatica*, 12:169–191, 2008.

[HS09] M. Held and C. Spielberger. A Smooth Spiral Tool Path for High Speed Machining of 2D Pockets. *Comput. Aided Design*, 41(7):539–550, July 2009.

[Ink] Inkscape. `http://inkscape.org/`. accessed on May, 2011.

[IST09] M. Ishaque, B. Speckmann, and C.D. Tóth. Shooting Permanent Rays among Disjoint Polygons in the Plane. In *Proc. 25th Annu. ACM Sympos. Comput. Geom.*, pages 51–60, Aarhus, Denmark, 2009.

[Kle89] R. Klein. *Concrete and Abstract Voronoi Diagrams*, volume 400 of *Lecture Notes in Computer Science*. Springer-Verlag, 1989. ISBN 3-540-52055-4.

[KLN09] R. Klein, E. Langetepe, and Z. Nilforoushan. Abstract Voronoi diagrams revisited. *Comput. Geom. Theory and Appl.*, 42(9):885 – 902, 2009.

[KMM93] R. Klein, K. Mehlhorn, and S. Meiser. Randomized Incremental Construction of Abstract Voronoi Diagrams. *Comput. Geom. Theory and Appl.*, 3(3):157–184, 1993.

[KW11] T. Kelly and P. Wonka. Interactive Architectural Modeling with Procedural Extrusions. *ACM Trans. Graph.*, 2011. accepted, not yet published.

[LD03] R. G. Laycock and A. M. Day. Automatically generating large urban environments based on the footprint data of buildings. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*, SM '03, pages 346–351, New York, NY, USA, 2003. ACM.

[MPF] The GNU MPFR Library. `http://www.mpfr.org/`. accessed on May, 2011.

[MWH+06] P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. Van Gool. Procedural Modeling of Buildings. *ACM Trans. Graph.*, 25:614–623, July 2006.

[OPC96] J.M. Oliva, M. Perrin, and S. Coquillart. 3D Reconstruction of Complex Polyhedral Shapes from Contours Using a Simplified Generalized Voronoi Diagram. *Comput. Graph. Forum*, 15(3):397–408, 1996.

[OSM] OpenStreetMap. `http://www.openstreetmap.org`. accessed on May, 2011.

[Pap98] E. Papadopoulou. L^∞ Voronoi Diagrams and Applications to VLSI Layout and Manufacturing. In *Proc. 9th Annu. Internat. Sympos. Algorithms Comput. (ISAAC*

'98), pages 9–18, London, UK, 1998. Springer-Verlag.

[PC03] S.C. Park and Y.C. Chung. Mitered offset for profile machining. *Comput. Aided Design*, 35(5):501–505, Apr 2003.

[Sha94] M. Sharir. Almost Tight Upper Bounds for Lower Envelopes in Higher Dimensions. *Discrete Comput. Geom.*, 12:327–345, 1994.

[TA09] M. Tănase-Avătavului. *Shape Decomposition and Retrieval*. PhD thesis, Universiteit Utrecht, Faculteit Wiskunde en Informatica, Utrecht, Netherlands, 2009.

[TV04a] M. Tănase and R. C. Veltkamp. A Straight Skeleton Approximating the Medial Axis. In *Proc. 12th Annu. Europ. Symp. Algorithms (ESA '04)*, pages 809–821, Bergen, Norway, Sep 2004.

[TV04b] M. Tănase and R.C. Veltkamp. Straight Line Skeleton in Linear Time, Topologically Equivalent to the Medial Axis. In *Proc. 20th Europ. Workshop Comput. Geom.*, Mar 2004.

[Vya09a] K. Vyatkina. Linear Axis for Planar Straight Line Graphs. In *Proc. of the 15th Australasian Symposium on Computing*, volume 94, pages 139–152, Darlinghurst, Australia, 2009. Australian Computer Society.

[Vya09b] K. Vyatkina. On the Structure of Straight Skeletons. In *Transactions on Computational Science VI*, volume 5730 of *Lecture Notes in Computer Science*, pages 362–379. Springer Berlin / Heidelberg, 2009.

[Yak04] E. Yakersberg. *Morphing Between Geometric Shapes Using Straight-Skeleton-Based Interpolation*. Msc thesis, Technion – Israel Institue of Technology, May 2004.

[Yap87] C.K. Yap. An $O(n \log n)$ Algorithm for the Voronoi Diagram of a Set of Simple Curve Segments. *Discrete Comput. Geom.*, 2(4):365–393, 1987.

[Yap04] C. K. Yap. Robust Geometric Computation. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 41, pages 927–952. Chapmen & Hall/CRC, Boca Raton, FL, 2 edition, 2004.

INDEX

Computing Straight Skeletons and Motorcycle Graphs: Theory and Practice

A B S T R A C T

摘要

The straight skeleton is a geometric structure that is similar to generalized Voronoi diagrams. Straight skeletons were introduced to the field of computational geometry one and a half decades ago. Since then many industrial and academical applications emerged, such as the computation of mitered offset curves, automatic roof construction, solving fold-and-cut problems and the reconstruction of surfaces, to name the most prominent ones. However, there is a significant gap between the most efficient straight-skeleton algorithms and implementations, on one hand, and the best known lower runtime bounds on the other hand. The primary goal of this thesis is the development of an algorithm that is suitable for implementation and efficient in terms of time and space in order to make the advantages of straight skeletons available to real-world applications.

直骨架是一种类似于广义Voronoi图的几何结构。直骨架在十多年前被引入到计算几何领域。自那时以来，涌现了许多工业和学术应用，例如计算斜接偏移曲线、自动屋顶建造、解决折叠切割问题以及表面重建，仅举几个最突出的例子。然而，在最有效的直骨架算法和实现，与已知的最佳运行时下界之间存在显著差距。本论文的主要目标是开发一种适用于实现，并且在时间和空间方面都高效的算法，以便使直骨架的优势能够应用于现实世界的应用。

We start with investigations concerning upper and lower bounds on the number of socalled flip events that occur in the triangulation-based straight-skeleton algorithm by Aichholzer and Aurenhammer. In particular, we prove the existence of Steiner triangulations that are free of flip events. This result motivates a novel straight-skeleton algorithm for non-degenerate simple polygons that is based on the so-called motorcycle graph. In order to extend this algorithm to arbitrary planar straight-line graphs, we carefully generalize the motorcycle graph. This generalization leads to practical and theoretical applications: Firstly, we obtain an extension of the alternative characterization of straight skeletons by Cheng and Vigneron to planar straight-line graphs. Secondly, this characterization motivates a straight-skeleton algorithm that is based on 3D graphics hardware. Thirdly, the generalized motorcycle graph leads to a wavefront-type straight-skeleton algorithm for arbitrary planar straight-line graphs. Our algorithm is easy to implement, has a theoretical worstcase time complexity of O(n2 log n) and operates in O(n) space. Extensive runtime tests with our implementation Bone exhibit an actual runtime of O(n log n) on a database containing more than 13 500 datasets of different characteristics. In practice, this constitutes an improvement of a linear factor in time and space compared to the current state-of-the-art straight-skeleton code, which is shipped

with the CGAL library. In particular, Bone performs up to 100 times faster than the current CGAL code on datasets with a few thousand vertices, requires significant less memory and accepts more general input.

我们首先研究由 Aichholzer 和 Aurenhammer 提出的基于三角剖分的直骨架算法中出现的所谓翻转事件的数量的上下界。特别地，我们证明了存在没有翻转事件的斯坦纳三角剖分。这一结果促使我们提出了一种基于所谓摩托车图的非退化简单多边形的新型直骨架算法。为了将此算法扩展到任意平面直线图，我们仔细地推广了摩托车图。这种推广带来了实际和理论应用：首先，我们将 Cheng 和 Vigneron 对直骨架的替代表征扩展到平面直线图。其次，这种表征激发了一种基于 3D 图形硬件的直骨架算法。第三，广义摩托车图产生了一种用于任意平面直线图的波前型直骨架算法。我们的算法易于实现，理论最坏情况时间复杂度为 O(n2 log n)，并在 O(n) 空间中运行。我们使用 Bone 实现的大量运行时测试表明，在一个包含超过 13 500 个不同特征数据集的数据库上，实际运行时为 O(n log n)。在实践中，与 CGAL 库附带的当前最先进的直骨架代码相比，这在时间和空间上都提高了线性因子。特别地，在具有数千个顶点的数据集上，Bone 的执行速度比当前的 CGAL 代码快 100 倍，需要的内存更少，并且接受更通用的输入。
Our straight-skeleton algorithm motivates the investigation of motorcycle graphs and their practical computation. We start with stochastic considerations of the average length of a motorcycles trace. The results obtained motivate a simple yet fast algorithm that employs geometric hashing. Runtime tests with our implementation Moca exhibit an O(n log n) runtime on the vast majority of our datasets. Finally, we revisit the geometric relation of straight skeletons and motorcycle graph. We present an algorithm that constructs planar straight-line graphs whose straight skeleton approximates any given motorcycle graph to an arbitrary precision. This algorithm finally leads to a P-completeness proof for straight skeletons of polygons with holes that is based on the P-completeness of motorcycle graphs.

我们的直线骨架算法激发了对摩托车图及其在实践中计算的研究。我们首先对摩托车轨迹的平均长度进行随机性考虑。所得结果促使我们采用一种简单而快速的算法，该算法采用了几何哈希。我们使用我们的实现Moca进行的运行时测试表明，在我们绝大多数数据集上，运行时复杂度为O(n log n)。最后，我们重新审视直线骨架和摩托车图的几何关系。我们提出了一种算法，该算法构建的平面直线图的直线骨架可以以任意精度逼近任何给定的摩托车图。该算法最终导出了一个关于带孔多边形直线骨架的P-完全性证明，该证明基于摩托车图的P-完全性。
I would like to thank my parents, Stefan and Gabriele, for their unlimited and unconditional support throughout my entire life. Special thanks go to my girlfriend Christina for her constant support and understanding during my doctoral studies.

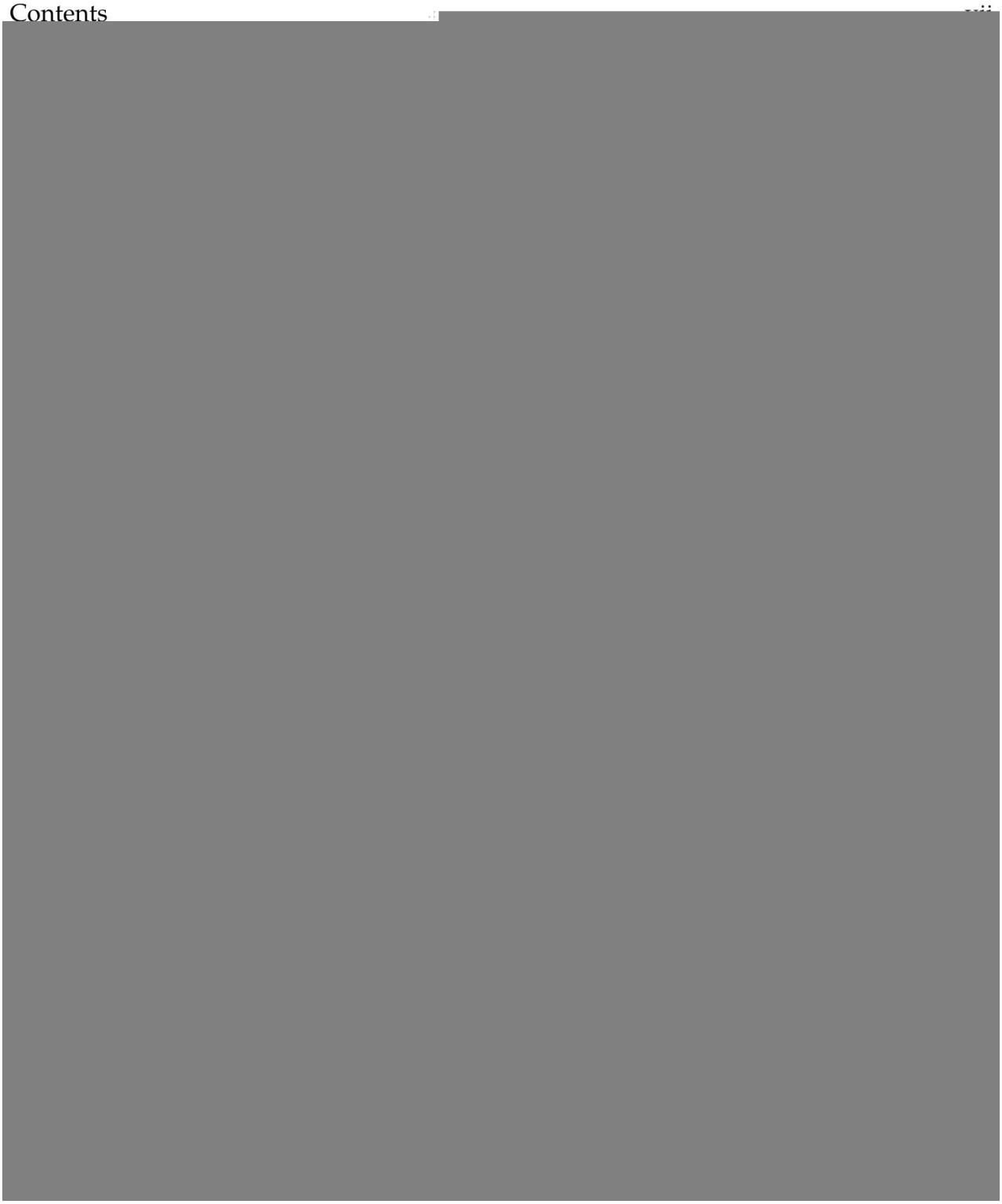
我要感谢我的父母，Stefan 和 Gabriele，感谢他们在我一生中给予我的无限和无条件的支持。特别感谢我的女朋友 Christina，感谢她在我的博士学习期间给予我的持续支持和理解。

I also thank my supervisor Martin Held for his guidance and his extensive support. Further, I thank my colleagues and co-workers. In particular I thank Gerhard Mitterlechner and Roland Kwitt for their proofreading and valuable discussions.

我还要感谢我的导师马丁·赫尔德（Martin Held）的指导和大力支持。此外，我感谢我的同事和合作者。特别感谢格哈德·米特勒赫纳（Gerhard Mitterlechner）和罗兰·奎特（Roland Kwitt）的校对和宝贵讨论。
Eventually, I thank the Austrian Science Fund (FWF) for supporting this thesis by funding project no. L367-N15.

最后，我感谢奥地利科学基金会（FWF）通过资助项目编号L367-N15来支持本论文。

Contents



Assume we are given a simple1 polygon in the plane which constitutes the outer walls of a house in a ground plan. How can we automatically build a roof such that all faces of the roof have equal slope and no rain drop gets caught in a sink? Assume we are given a polygon with holes2 which is to be milled out with a numerically controlled machine (NC machine). How do we compute so-called offset curves of the input such that sharp vertices of the input remain sharp for the offset curve?3 Assume we are given a simple polygon drawn on a paper. How can we determine a series of complete folds of the paper and a subsequent single cut through the folded paper such that one of the resulting paper pieces has the shape of the polygon? These three problems share a common feature: they can be solved using the straight skeleton.

假设我们有一个平面上的简单多边形simple1，它构成了房屋地基平面的外墙。如何自动构建一个屋顶，使得屋顶的所有面都具有相同的坡度，并且没有雨滴会被困在凹陷处？假设我们有一个带孔的多边形holes2，需要用数控机床（NC机床）铣削出来。我们如何计算输入的所谓偏移曲线，使得输入的尖锐顶点对于偏移曲线保持尖锐？3假设我们有一个画在纸

上的简单多边形。我们如何确定一系列完整的纸张折叠方式，以及随后通过折叠纸张的单次切割，使得其中一个产生的纸片具有该多边形的形状？这三个问题有一个共同的特点：它们可以使用直线骨架来解决。
Roughly speaking, the straight skeleton of a simple polygon P is a tree-like skeleton structure that is similar to Voronoi diagrams of polygons, but consists of straight-line segments only. The definition of the straight skeleton is based on the so-called wavefront propagation process. In a nutshell, the straight skeleton of P consists of the set of points that are traced out by the vertices of P when the edges of P shrink inwards in a self-parallel manner and with constant speed. During this propagation process, some edges may vanish and the polygon may successively disintegrate into multiple parts, see Figure 1. Straight skeletons of simple polygons were introduced by Aichholzer et al. [AAAG95] and were subsequently generalized to planar straight-line graphs4 by Aichholzer and Aurenhammer [AA96] about one and a half decades ago. Similar to Voronoi diagrams, straight skeletons possess a heterogeneous plenitude of applications and many of them appeared just in the past decade. The list of applications includes

粗略地说，简单多边形P的直线骨架是一种树状骨架结构，类似于多边形的Voronoi图，但仅由直线段组成。直线骨架的定义基于所谓的波前传播过程。简而言之，P的直线骨架由P的顶点在P的边以自平行方式和恒定速度向内收缩时所描绘出的一组点组成。在此传播过程中，某些边可能会消失，并且多边形可能会连续分解成多个部分，参见图1。简单多边形的直线骨架由Aichholzer等人[AAAG95]引入，随后由Aichholzer和Aurenhammer [AA96]在大约十五年前推广到平面直线图4。与Voronoi图类似，直线骨架拥有各种各样的应用，其中许多应用仅在过去十年中出现。应用列表包括

• the computation of mitered offset curves and motion planning in computer-aided design and computer-aided manufacturing (CAD/CAM) [PC03];

• 计算机辅助设计和计算机辅助制造 (CAD/CAM) 中斜接偏移曲线的计算和运动规划 [PC03]；

• computing the quickest walking paths in a Manhattan-style city with a fast public transit by means of the city Voronoi diagram; computing critical areas in VLSI circuit models [AAP04, Pap98];

• 通过城市Voronoi图计算曼哈顿式城市中利用快速公共交通的最快步行路径；计算VLSI电路模型中的关键区域[AAP04, Pap98]；

• automatic roof generation, terrain modeling and area collapsing within maps in geographical information systems (GIS) [AAAG95, AA96, LD03, MWH+06, KW11, Hav05, HS08];

• 地理信息系统（GIS）中地图内的自动屋顶生成、地形建模和区域坍塌 [AAAG95, AA96, LD03, MWH+06, KW11, Hav05, HS08];
• and polygon decompositions [TV04b].

• 和多边形分解 [TV04b]。

• solving fold-and-cut problems, computing hinged dissections of polyhedra and related problems in the field of mathematical origami [DO07, DDL98, DDM00, DDLS05];

解决折叠切割问题，计算多面体的铰接剖分以及数学折纸领域中的相关问题[DO07, DDL98, DDM00, DDLS05]；

• shape reconstruction and interpolation of contour lines in three-dimensional space [OPC96, BGLSS04];

• 三维空间中轮廓线的形状重构和插值 [OPC96, BGLSS04];

Some of these applications are genuine to straight skeletons, like fold-and-cut problems or automatic roof construction. Other applications are inherited from generalizations of Voronoi diagrams, like many problems in the fields of CAD/CAM, VLSI and GIS.

这些应用中有一些是直线骨架所特有的，例如折叠切割问题或自动屋顶建造。其他的应用则继承自Voronoi图的推广，例如CAD/CAM、VLSI和GIS等领域的许多问题。

In contrast to the large number of applications, we perceive a gap between available algorithms and implementations on the one hand and the best known lower time bound on the other hand. At the moment, the best known lower time bound for the computation of straight skeletons is Ω(n) for an n-vertex polygon and Ω(n log n) for an n-vertex planar straight line graph. However, the currently fastest algorithm for arbitrary polygons and planar straight-line graphs is due to Eppstein and Erickson [EE99] and has a theoretical worst-case time complexity of O(n17/11+ϵ), where ϵ > 0 is an arbitrary small number. The only straight-skeleton implementation available is shipped with the CGAL library [CGA] and was implemented by Cacciola [Cac04]. However, our experiments revealed that the underlying algorithm needs O(n2 log n) time and O(n2) space for practical input data.

与大量的应用形成对比的是，我们注意到现有算法和实现一方面与已知的最佳时间下界之间存在差距。目前，对于一个n个顶点的多边形，计算直线骨架的最佳已知时间下界是Ω(n)；对于一个n个顶点的平面直线图，该下界是Ω(n log n)。然而，目前针对任意多边形和平面直线图的最快算法归功于Eppstein和Erickson [EE99]，其理论最坏情况时间复杂度为O(n17/11+ϵ)，其中ε > 0是一个任意小的数。唯一可用的直线骨架实现随CGAL库[CGA]一起发布，由Cacciola [Cac04]实现。然而，我们的实验表明，对于实际的输入数据，底层算法需要O(n2 log n)的时间和O(n2)的空间。

This thesis deals with theoretical properties and the practical computation of straight skeletons and motorcycle graphs. The goal is to develop a practice-minded straight-skeleton algorithm, which accepts arbitrary planar straight-line graphs as input, is efficient in time and space and easy to implement.

本论文探讨了直线骨架和摩托车图的理论性质和实际计算。目标是开发一种注重实践的直线骨架算法，该算法接受任意平面直线图作为输入，在时间和空间上都高效且易于实现。

1.1 o r g a n i z a t i o n

1.1 组织结构
In this chapter, we start with the definition of straight skeletons, the terrain model and the motorcycle graph in Section 1.2. We continue with a presentation of several applications of straight skeletons in Section 1.3. In Section 1.4, we review known algorithms and implementations of straight skeletons and motorcycle graphs and in Section 1.5, we discuss different approaches by which straight skeletons were generalized.

在本章中，我们首先在第1.2节中给出直线骨架、地形模型和摩托车图的定义。然后，在第1.3节中介绍直线骨架的几个应用。在第1.4节中，我们回顾已知的直线骨架和摩托车图的算法和实现，并在第1.5节中，我们讨论直线骨架推广的不同方法。

Chapter 2 is devoted to the computation of straight skeletons. In Section 2.1, we collect known geometric properties of the straight skeleton and present them along with their proofs in a unified formalism based on the definitions in Section 1.2. In Section 2.2, we analyze the triangulation-based algorithm by Aichholzer and Aurenhammer [AA98]. We prove a few results concerning the lower and upper bounds for the number of the so-called flip events and show that Steiner triangulations exist where no flip events occur.

第2章致力于直线骨架的计算。在2.1节中，我们收集了直线骨架的已知几何性质，并根据1.2节中的定义，以统一的形式呈现它们及其证明。在2.2节中，我们分析了Aichholzer和Aurenhammer [AA98]提出的基于三角剖分的算法。我们证明了一些关于翻转事件数量的上下界的结果，并表明存在没有发生翻转事件的Steiner三角剖分。

This insight motivates a novel straight-skeleton algorithm for simple non-degenerate5 polygons that is based on the motorcycle graph in Section 2.3. In order to extend this approach to arbitrary planar straight-line graphs, we carefully generalize the motorcycle graph in Section 2.4. Further, we prove two essential geometric properties for this generalization that are important for our algorithmic approach: (i) the input graph and the motorcycle

这一见解促使我们提出了一种新的针对简单非退化5多边形的直骨架算法，该算法基于第2.3节中的摩托车图。为了将这种方法扩展到任意平面直线图，我们仔细地推广了第2.4节中的摩托车图。此外，我们证明了这种推广的两个基本几何性质，这对我们的算法方法非常重要：（i）输入图和摩托车

Preliminaries and Definitions - 靠岸学术

1.2 p r e l i m i n a r i e s a n d d e f i n i t i o n s

1.2 预备知识与定义

1.2.1 The straight skeleton of a simple polygon

1.2.1 简单多边形的直骨架

Aichholzer et al. [AAAG95] introduced the straight skeleton of simple polygons P by considering a so-called wavefront-propagation process. Each edge e of P sends out a wavefront which moves inwards at unit speed and is parallel to e. The wavefront of P can be thought to shrink in a self-parallel manner such that sharp corners at reflex6 vertices of P remain sharp, see Figure 1. During this propagation process topological changes, so-called events, will occur: Edges collapse to zero length and the wavefront may be split into multiple parts. Each wavefront vertex moves along the bisector of two edges of P and, while it moves, it traces out a straight-line segment.

Aichholzer等人[AAAG95]通过考虑所谓的波前传播过程，引入了简单多边形P的直线骨架。P的每条边e都会发出一个波前，该波前以单位速度向内移动，并且平行于e。P的波前可以被认为是以自平行的方式收缩，使得P的凹顶点处的尖角保持尖锐，参见图1。在这个传播过程中，会发生拓扑变化，即所谓的事件：边塌缩为零长度，并且波前可能分裂成多个部分。每个波前顶点沿着P的两条边的平分线移动，并且在移动时，它会描绘出一条直线段。

Definition 1.1 (straight skeleton, arcs, nodes, faces). The straight skeleton S (P) of the polygon P comprises the straight-line segments that are traced out by the wavefront vertices.

定义 1.1（直骨架，弧，节点，面）[NT0]。多边形 P 的直骨架 S (P) 包含由波前顶点描绘出的直线段。

These straight-line segments and are called the arcs of S (P). The loci of the topological changes of the wavefront are called nodes. To each edge e of P belongs a face f (e), which consists of all points that are swept by the wavefront edge that is sent out by e.

这些直线段被称为S(P)的弧。波前拓扑变化的轨迹被称为节点。对于P的每条边e，都对应一个面f(e)，它由波前边缘扫过的所有点组成，该波前边缘由e发出。

We illustrate the straight skeleton S (P) of a simple polygon P in Figure 1. Each node is incident to multiple arcs. The boundary of a face consists of arcs and the vertices of a face are nodes.

我们在图1中展示了一个简单多边形P的直骨架S (P)。每个节点都与多个弧相关联。一个面的边界由弧组成，一个面的顶点是节点。
Lemma 1.2 ([AAAG95]). The straight skeleton S (P) of a simple polygon P with n vertices is a tree. It consists n − 2 nodes and 2n − 3 arcs and tessellates P into n faces.

引理 1.2 ([AAAG95]).简单多边形P的直线骨架S(P)是一棵树。它由n − 2个节点和2n − 3条弧组成，并将P细分为n个面。

For the proof of this lemma we create a single node for each topological change, even if the resulting nodes coincide geometrically. In particular, we assume that each node has degree three.[7] For example, if P is a regular n-gon then S (P) has a star form, i. e. all arcs meet in the center point of P and multiple nodes with degree three are geometrically coincident.

为了证明这个引理，我们为每个拓扑变化创建一个单独的节点，即使生成的节点在几何上重合。特别地，我们假设每个节点的度数为三。[7] 例如，如果 P 是一个正n边形，那么 S (P) 具有星形形式，即所有弧都相交于 P 的中心点，并且多个度数为三的节点在几何上重合。

Proof. The number of faces is n, since P comprises n edges. Next, we note that the face f (e) of an edge e is connected because it is given by the set of points which are swept by a continuously moving wavefront edge. On the other hand, each point within P is reached by the wavefront after some time. Hence, P is tessellated by the faces of S (P) and S (P) consists of the boundaries of the faces. From that it follows that S (P) does not contain a cycle and is therefore a tree. The inner nodes of this tree are of degree 3 and the tree has n leaves. It follows that S (P) has n − 2 nodes and 2n − 3 arcs.

证明。面的数量为n，因为P包含n条边。接下来，我们注意到边e的面f (e)是连通的，因为它是由连续移动的波前边扫过的点集给出的。另一方面，P内的每个点在一段时间后都会被波前到达。因此，P被S (P)的面镶嵌，并且S (P)由这些面的边界组成。由此可知，S (P)不包含环，因此是一棵树。这棵树的内部节点度数为3，并且这棵树有n个叶子。由此得出，S (P)有n − 2个节点和2n − 3条弧。

Definition 1.3 (reflex/convex wavefront vertex). A wavefront vertex v is reflex if the angle on the side where v propagates to is at least 180°. Likewise, we define a convex wavefront vertex.

定义 1.3（反射/凸波前顶点）[NT0]。如果 v 传播侧的角度至少为 180[NT3]°，则波前顶点 v 是反射的。同样，我们定义一个凸波前顶点。

Let us discuss the different types of topological events that occur for the wavefront in more detail. We follow Aichholzer et al. [AAAG95], who distinguish two types of events, namely edge events and split events.

让我们更详细地讨论波前发生的各种类型的拓扑事件。我们遵循 Aichholzer 等人 [AAAG95] 的研究，他们区分了两种类型的事件，即边事件和分裂事件。

• Edge event: An edge event occurs when two neighboring convex vertices u and v of the wavefront meet. This event causes the wavefront edge e, which connects u and v, to collapse to zero length. The wavefront edge e is removed and the vertices u and v are merged into a new convex vertex with its own velocity, see Figure 2 (a).

• 边事件：当波前的两个相邻凸顶点 u 和 v 相遇时，会发生边事件。此事件导致连接 u 和 v 的波前边 e 塌陷为零长度。波前边 e 被移除，顶点 u 和 v 合并为一个具有自身速度的新凸顶点，参见图 2 (a)。

• Split event: A split event occurs when a reflex vertex u of the wavefront meets an edge e of the wavefront. The vertex u splits the entire wavefront into polygonal parts. Each part keeps on propagating for its own, see Figure 2 (b).

• 分裂事件：当波前的一个反射顶点 u 遇到波前的一条边 e 时，会发生分裂事件。顶点 u 将整个波前分裂成多边形部分。每个部分各自继续传播，参见图 2 (b)。
Split events can occur in interesting variations. In Figure 2 (c), a split event for the reflex vertex u and the edge e occurs, while at the same time the edge between u and v collapses to zero length. Note that in this case only one wavefront part remains after the split event. However, the question arises whether we would like to call this event an edge event as well. For subfigure (a) we observe that a local disk around the

分裂事件可能以有趣的变体形式发生。在图2(c)中，发生了反射顶点u和边e的分裂事件，同时u和v之间的边塌缩为零长度。请注意，在这种情况下，分裂事件后仅保留一个波前部分。然而，问题出现了，我们是否也想将此事件称为边事件。对于子图(a)，我们观察到围绕着一个局部圆盘

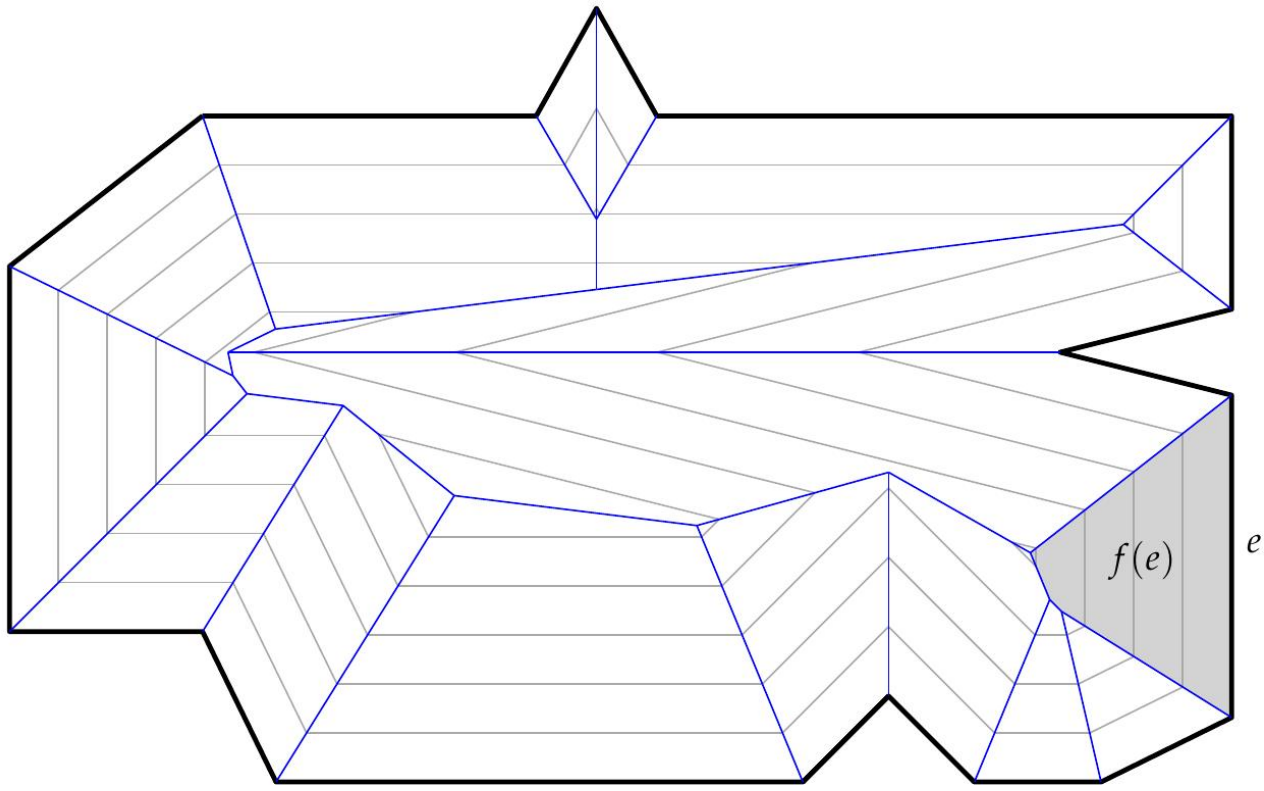


Figure 1: The straight skeleton S(P) (blue) of a simple polygon P (bold) is defined by a wavefront propagation process where the edges of P move inwards in a self-parallel manner. Five wavefronts at equally spaced points in time are shown in gray. The blue straight-line segments are called the arcs of S(P) and the common endpoints of arcs are called the nodes of S(P). We shade the face f (e) of one edge e in light gray.

图 1：简单多边形 P（粗体）的直骨架 S(P)（蓝色）由波前传播过程定义，其中 P 的边以自平行的方式向内移动。以等间隔的时间点显示的五个波前以灰色显示。蓝色的直线段称为 S(P) 的弧，弧的公共端点称为 S(P) 的节点。我们用浅灰色阴影表示一条边 e 的面 f (e)。

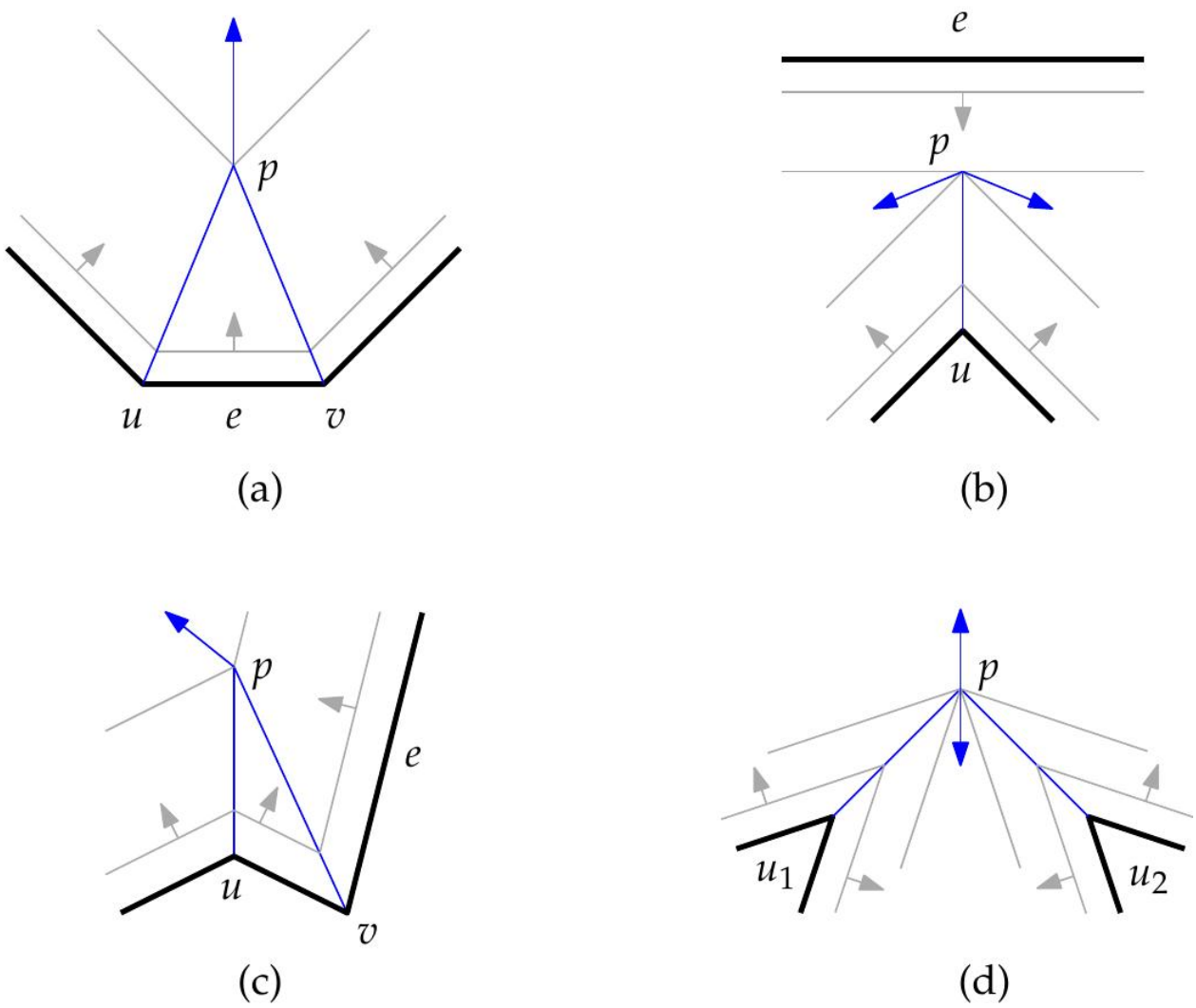


Figure 2: Two types of events occur during the wavefront propagation: edge and split events. (a) illustrates an edge event, (b) a simple split event, (c) a split event with an edge collapse combined, (d) a split event with two reflex vertices u1 and u2 involved. The arcs are depicted in blue and the wavefronts in gray.

图 2：波前传播过程中发生两种类型的事件：边缘事件和分裂事件。(a) 说明了一个边缘事件，(b) 一个简单的分裂事件，(c) 一个边缘塌陷结合的分裂事件，(d) 一个涉及两个反射顶点 u1 和 u2 的分裂事件。弧线用蓝色描绘，波前用灰色描绘。

point p, where the event happened, is tessellated into convex slices by the arcs of the straight skeleton. It is easy to see that this holds in general if u and v are convex. On the other hand, in subfigures (b) and (c), a tessellation of a local disk around p also contains a reflex slice. We would like to use this property as an indicator that a split event happened at p. For this reason, we call the event illustrated by subfigure (c) a split event.[8] As a consequence, edge events describe edge collapses between two convex vertices. Furthermore, if the straight-skeleton face of an edge — interpreted as a polygon — contains a reflex vertex, it has been generated by a split event.

事件发生的点p被直线骨架的弧线分割成凸切片。容易看出，如果u和v是凸的，则通常情况下这都成立。另一方面，在子图(b)和(c)中，围绕p的局部圆盘的镶嵌也包含一个凹切片。我们希望使用此属性作为在p处发生分割事件的指标。因此，我们将子图(c)所示的事件称为分割事件。[8] 因此，边事件描述了两个凸顶点之间的边塌陷。此外，如果一条边的直线骨架面（解释为多边形）包含一个凹顶点，则它是由分割事件生成的。

In subfigures (b) and (c) only one reflex vertex u was involved in the split event. However, it could happen that two or more reflex wavefront vertices u1, . . . , uk meet each other in a common point p at the same time, see Figure 2 (d). Such situations occur easily for rectilinear[9] polygons. As a consequence the wavefront is split into k parts. Note that it is possible that a new reflex vertex emerges, as illustrated in Figure 2 (d). We will call events, where two or more reflex vertices meet simultaneously in a point a multi split event. Multi split events that cause a new reflex vertex to emerge are called vertex events.[10]

在子图（b）和（c）中，只有一个凹顶点u参与了分裂事件。然而，可能会发生两个或多个凹波前顶点u1, . . . , uk在同一点p同时相遇的情况，参见图2（d）。对于直线[9]多边形，这种情况很容易发生。因此，波前被分成k部分。请注意，可能会出现一个新的凹顶点，如图2（d）所示。我们将两个或多个凹顶点同时在一个点相遇的事件称为多重分裂事件。导致新的凹顶点出现的多重分裂事件称为顶点事件。[10]

We want to remark that the taxonomy of the different classes of events is in flux within the literature of straight skeletons. For example, Tanaseˇ [TA09] pursues a different approach by considering three sub-classes of edge events and three sub-classes of split events. Following this taxonomy, the situation in Figure 2 (c) would be called reflex edge annihilation.

我们想指出的是，在直线骨架的文献中，不同事件类别的分类正在不断变化。例如，Tanaseˇ [TA09] 采用了一种不同的方法，考虑了边缘事件的三个子类和分裂事件的三个子类。按照这种分类方法，图 2 (c) 中的情况将被称为反射边缘湮灭。

Using the taxonomy we presented above, the propagation of a wavefront always proceeds as follows: Every edge event reduces the number of wavefront edges by one. A split event splits the wavefront into polygonal parts and implies a hierarchy of nested polygons. Further, a split event reduces the number of reflex vertices at least by one. Even if a vertex event happens, at least two reflex vertices must have been involved in this event. Eventually, all nested parts in the hierarchy become convex polygons. The

convex polygons are reduced by a series of edge events. Finally, a polygon vanishes by collapsing either to a point or to a straight-line segment. As an example for the first case one could imagine P to be a triangle, and for the second case a non-equilateral rectangle.

使用我们上面提出的分类法，波前的传播总是按照以下方式进行：每个边事件都会使波前边的数量减少一个。分裂事件将波前分裂成多边形部分，并暗示着一个嵌套多边形的层级结构。此外，分裂事件至少会使凹顶点（reflex vertices）的数量减少一个。即使发生顶点事件，至少有两个凹顶点必须参与到此事件中。最终，层级结构中的所有嵌套部分都变成凸多边形。凸多边形通过一系列边事件被简化。最后，一个多边形通过坍缩成一个点或一条直线段而消失。作为第一种情况的例子，可以想象P是一个三角形，而对于第二种情况，则是一个非等边矩形。

1.2.2 The straight skeleton of a planar straight-line graph

1.2.2 平面直线图的直线骨架

Aichholzer and Aurenhammer [AA96] generalized the concept of straight skeletons to planar straight-line graphs.[11] Assume we are given a planar straight-line graph G with no isolated12 vertices. The basic idea to define the straight skeleton S (G) of G remains the same: Every edge e of G sends out two wavefront edges — one at each side of e — which travel at unit speed and stay parallel to the original edge e. However, the terminal13 vertices of G play an interesting role, since it is a-priori unclear how the wavefront is expected to behave at such vertices. Aichholzer and Aurenhammer [AA96] decided that each terminal vertex v sends out a wavefront for it own, which is perpendicular to the single incident edge of v. As a result, the wavefront in the neighborhood of a terminal vertex v forms a rectangular cap around v, see Figure 3.

Aichholzer和Aurenhammer [AA96] 将直线骨架的概念推广到平面直线图。[11] 假设我们给定一个没有孤立顶点12的平面直线图G。定义G的直线骨架S(G)的基本思想保持不变：e发出两条波前边——e的每一侧各一条——它们以单位速度移动，并保持与原始边e平行。然而，G的终端13顶点扮演着一个有趣的角色，因为事先不清楚波前在这些顶点处应该如何表现。Aichholzer和Aurenhammer [AA96] 决定每个终端顶点v为其自身发出一个波前，该波前垂直于v的单条入射边。因此，在终端顶点v附近的波前围绕v形成一个矩形帽，参见图3。

The terminology of arcs and nodes remains the same in this general setting. We still have the two types of events — edge events and split events — which occur during the propagation of the wavefronts and we define S (G) as the set of points that are traced out by the vertices of the wavefront. Note that a vertex v of G with degree $k \geq 2$ causes k wavefront vertices to emanate from v. Furthermore, every terminal vertex v causes two reflex wavefront vertices to emanate, see Figure 3.

在这个一般设置中，弧和节点的术语保持不变。我们仍然有两种类型的事件——边事件和分裂事件——它们发生在波前传播期间，我们将 S (G) 定义为由波前顶点描绘出的点的集合。请注意，G 中度数为 $k \geq 2$ 的顶点 v 会导致 k 个波前顶点从 v 发出。此外，每个终端顶点 v 都会导致两个反射波前顶点发出，参见图 3。

Definition 1.4 (wavefront). We denote by W (G, t) the wavefront of a planar straight-line graph G at some time t ≥ 0.

定义 1.4 (波前)[NT0]。我们用 W (G, t) 表示平面直线图 G 在某个时间 t ≥ 0 的波前。

We interpret W (G, t) for a fixed t as a 2-regular graph. A topological event happens when the planarity of W (G, t) is violated, including the case that two wavefront vertices meet. We interpret W (G, 0) as the wavefront at time zero. While W (G, 0) is geometrically overlapping with G, we assume that W (G, 0) has — as a graph — the same topology as W (G, ϵ) for a small ϵ > 0. We observe that S (G) and G tessellate the plane into faces and that every face f (e) belongs to an wavefront edge e: The face consists of all points in the plane which are swept by e, see Figure 3.

对于固定的t，我们将W (G, t)解释为一个2-正则图。当W (G, t)的平面性被破坏时，会发生拓扑事件，包括两个波前顶点相遇的情况。我们将W (G, 0)解释为时间零时的波前。虽然W (G, 0)在几何上与G重叠，但我们假设W (G, 0)作为一个图，具有与W (G, ε)相同的拓扑结构，其中ε > 0是一个很小的数。我们观察到S (G)和G将平面分割成面，并且每个面f (e)属于一个波前边e：该面由平面上所有被e扫过的点组成，参见图3。

unbounded faces and infinite arcs Note that certain faces will be unbounded since every point in the plane is eventually reached by the wavefront. By interpreting S (G) as a graph, it is reasonable to demand that each face corresponds to a cycle in S (G). Let us consider the last topological event that happened to the wavefront. After this event, the wavefront is a cycle that circumscribes G and all bounded faces of S (G). The vertices of this cycle trace the infinite arcs of S (G). Topologically, we add this cycle to S (G) and add the corresponding “infinite” arc to each “infinite” vertex of the cycle. Each “infinite” vertex has degree three. This interpretation of S (G) has practical advantages, e. g., when computing offset curves based on the straight skeleton, see Section 1.3.1.

无界面和无限弧。请注意，某些面将是无界的，因为平面中的每个点最终都会被波前到达。通过将 S (G) 解释为一个图，可以合理地要求每个面对应于 S (G) 中的一个循环。让我们考虑波前发生的最后一个拓扑事件。在此事件之后，波前是一个环，它包围着 G 和 S (G) 的所有有界面。这个环的顶点追踪 S (G) 的无限弧。在拓扑上，我们将这个环添加到 S (G)，并将相应的“无限”弧添加到环的每个“无限”顶点。每个“无限”顶点都有三度。S (G) 的这种解释具有实际优势，例如，在计算基于直线骨架的偏移曲线时，请参见第 1.3.1 节。

isolated vertices The definition of S (G) presented above assumes that G does not contain an isolated vertex. Aichholzer and Aurenhammer [AA96] mentioned that an isolated vertex could be approximated by a small straight-line segment. This must be done with some caution since S (G) does not continuously depend on G. For example, when multiple reflex arcs are incident to a point then a small perturbation of G would lead to a dramatically different straight skeleton S (G). See more on this issue in Section 2.5.3. However, one could simply define that an isolated vertex v emanates a wavefront which forms an axis-aligned square or any other polygon that has the property that its edges have an orthogonal distance of ϵ at time ϵ. That is, the supporting lines of the wavefront edges at time ϵ are tangential to the disk centered at v with radius ϵ. In fact,

Demaine and O'Rourke [DO07] define the wavefronts emanated by an isolated vertex as an axis-aligned square in order to apply the straight skeleton for their fold-and-cut problems, see Section 1.3.3.

孤立顶点。上面给出的 S (G) 的定义假设 G 不包含孤立顶点。Aichholzer 和 Aurenhammer [AA96] 提到，孤立顶点可以用一个小的直线段来近似。必须谨慎地进行此操作，因为 S (G) 并不连续地依赖于 G。例如，当多个反射弧与一个点相交时，G 的一个小扰动会导致截然不同的直线骨架 S (G)。有关此问题的更多信息，请参见第 2.5.3 节。但是，可以简单地定义一个孤立顶点 v 发出波前，该波前形成一个轴对齐的正方形或任何其他具有以下属性的多边形：其边在时间 ϵ 处具有 ϵ 的正交距离。也就是说，时间 ϵ 时波前边的支撑线与以 v 为中心、半径为 ϵ 的圆盘相切。事实上，Demaine 和 O'Rourke [DO07] 将孤立顶点发出的波前定义为轴对齐的正方形，以便将直线骨架应用于他们的折叠和切割问题，参见第 1.3.3 节。
terminal vertices In the same manner we could also alter the wavefront shape around terminal vertices. Consider a terminal vertex v and the two wavefront edges that emanate from the single incident edge e. Then we could basically place an arbitrary polygonal cap around v which connects the two wavefront edges from e. We only require that the segments of the cap have orthogonal distance ϵ at time ϵ. Nevertheless, using a rectangular cap — as introduced by Aichholzer and Aurenhammer — appears to be the most natural approach.

终端顶点 同样地，我们也可以改变终端顶点周围的波前形状。考虑一个终端顶点v和从单条入射边e发出的两条波前边。然后，我们基本上可以在v周围放置一个任意多边形帽，将来自e的两条波前边连接起来。我们仅要求帽的线段在时间ϵ处具有正交距离ϵ。然而，使用由Aichholzer和Aurenhammer引入的矩形帽似乎是最自然的方法。

1.2.3 Roof and terrain model

1.2.3 屋顶和地形模型

Aichholzer et al. [AAAG95] presented an interpretation for the straight skeleton of simple polygons, which was extended to planar straight-line graphs by Aichholzer and Aurenhammer [AA96]. It turns out that this interpretation is a versatile tool in proofs of geometric properties of the straight skeleton. Further, it leads to one of the prominent applications of straight skeletons: roof construction and terrain modeling, see Section 1.3.2. The basic idea is to embed the wavefront propagation process in R3 in the following sense: the first two dimensions represent the plane spatial dimensions and the third dimension reflects the temporal dimension. The wavefront propagation now defines the so-called terrain T (G) of G as follows.

Aichholzer等人 [AAAG95] 提出了对简单多边形直骨架的一种解释，Aichholzer和 Aurenhammer [AA96] 将其扩展到平面直线图。事实证明，这种解释是证明直骨架几何性质的一种通用工具。此外，它引出了直骨架的一个重要应用：屋顶建造和地形建模，参见第1.3.2节。基本思想是将波前传播过程嵌入到R3中，其含义如下：前两个维度表示平面空间维度，第三个维度反映时间维度。波前传播现在将G的地形定义为所谓的T (G)。

Definition 1.5 (terrain). The terrain T (G) of G is defined by

定义 1.5 (地形)[NT0]。G 的地形 T (G) 定义为

$$T(G) := = [t \geq 0 W(G, t) \times \{t\}$$

▪

(1.1)
Figure 4 illustrates the terrain T (G) of the graph G that is illustrated in Figure 3. Aichholzer et al. [AAAG95] used the term roof resp. island to indicate that T (P) of a simple polygon P has the following two interpretations. Firstly, T (P) can be interpreted as a particular roof of a house for which P models the footprint of the outer walls. Secondly, one can interpret P as the coastline of an island that has the shape of T (P). If the surrounding sea floods the island then the rising coastline has the shape of the rising wavefront in R3. In the case of planar straight-line graphs G Aichholzer and Aurenhammer [AA96] use the term terrain for T (G).

图 4 说明了图 G 的地形 T (G)，该图在图 3 中进行了说明。Aichholzer 等人 [AAAG95] 使用了术语 roof resp.island 来表示简单多边形 P 的 T (P) 具有以下两种解释。首先，T (P) 可以解释为房屋的特定屋顶，其中 P 模拟了外墙的足迹。其次，可以将 P 解释为具有 T (P) 形状的岛屿的海岸线。如果周围的海水淹没了该岛屿，那么上升的海岸线就具有 R3 中上升波前的形状。在平面直线图 G 的情况下，Aichholzer 和 Aurenhammer [AA96] 使用术语 terrain 表示 T (G)。

The terrain T (G) consists of plane facets that have a slope identical to the inverse of the propagation speed of the wavefront edges, which is 1. An edge of T (G) can either be convex or reflex. In the ordinary sense, we call an edge e of T (G) convex if the intersection of a small disk at any point in the relative interior of e with the points below T (G) is always convex. A reflex edge e of T (G) is defined likewise.

地形T(G)由平面组成，这些平面具有与波前边缘传播速度的倒数相同的坡度，即1。T(G)的边可以是凸边或凹边。通常意义上，如果位于e相对内部的任何一点上的小圆盘与T(G)下方点的交集始终是凸的，则我们称T(G)的边e为凸边。T(G)的凹边e的定义类似。

Definition 1.6 (reflex/convex arc, valley, ridge). We call the arcs of S (G) which are traced out by reflex (convex) wavefront vertices reflex arcs (convex arcs). We call a reflex edge of T (G) a valley and a convex edge of T (G) a ridge.

定义 1.6（凹/凸弧，谷，脊）[NT0]。我们称由凹（凸）波前顶点描绘出的S (G)的弧为凹弧（凸弧）。我们称T (G)的凹边为谷，T (G)的凸边为脊。

Observation 1.7 ([AAAG95, AA98]). The straight skeleton S (G) is the projection of the valleys and ridges of T (G) onto the plane R2 × {0}. Moreover, the valleys correspond to the reflex arcs and the ridges correspond to the convex arcs.

观察 1.7 ([AAAG95, AA98]).直线骨架 S (G) 是 T (G) 的谷和脊在平面 R2 × {0} 上的投影。此外，谷对应于凹弧，脊对应于凸弧。

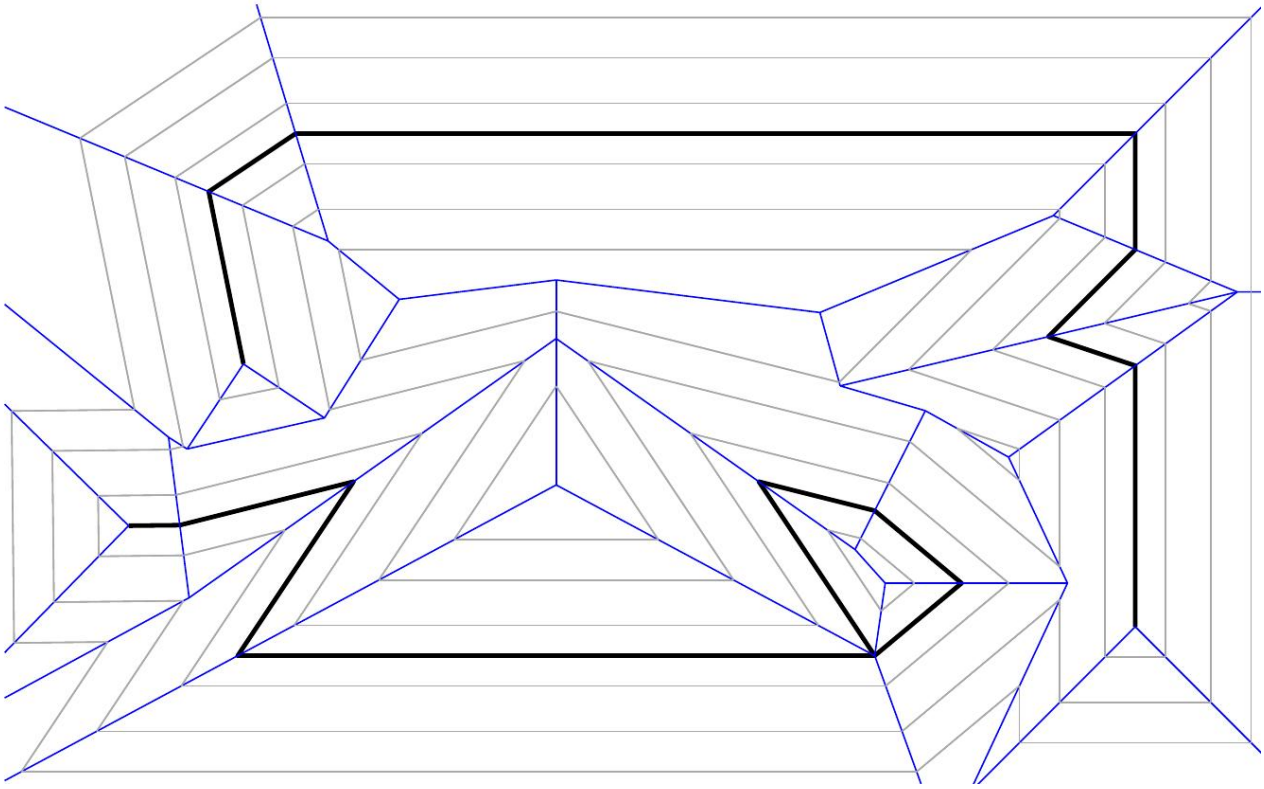


Figure 3: The straight skeleton S(G) (blue) of the planar straight-line graph G (bold). The wavefronts at three points in time are depicted in gray.

图3：平面直线图G（粗体）的直骨架S(G)（蓝色）。以灰色描绘了三个时间点的波前。

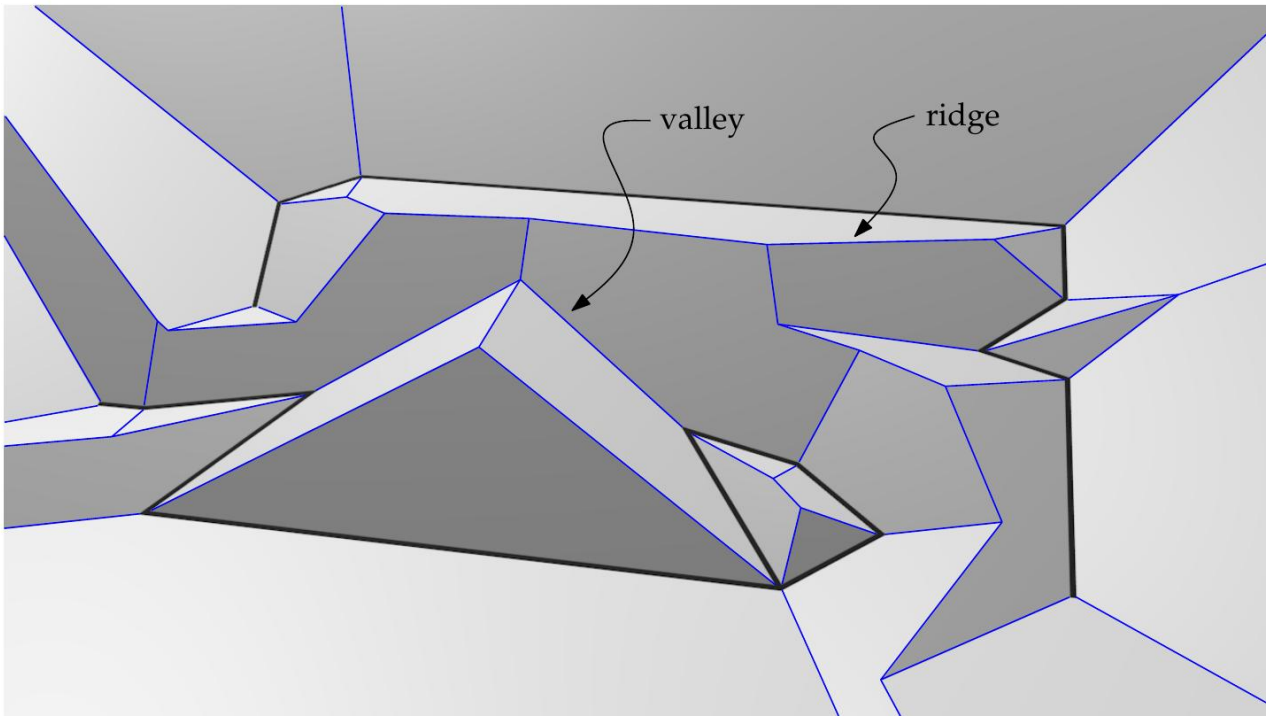


Figure 4: The terrain T (G) of the graph G which is illustrated in Figure 3. The ridges and valleys are in blue.

图 4：图 G 的地形 T (G)，如图 3 所示。山脊和山谷以蓝色表示。

roofs of polygons Aichholzer et al. [AAAG95] discussed the roof model of simple polygons P in more detail. They investigated more general roofs R on P which fulfill the property that each facet lies on a plane that contains an edge of P and has slope 1. The question arises whether such an R is equal to T (P). It turns out that this is not necessarily the case. However, R and T (P) are equal if all valleys of R are incident to P, or alternatively, if for any point x ∈ R the path of the steepest descent leads to P. In other words, Aichholzer et al. [AAAG95] showed that among all roofs, T (P) has the peculiar property that it does not accumulate water when it is raining.

Aichholzer等人[AAAG95]讨论了简单多边形的屋顶模型P，并进行了更详细的阐述。他们研究了P上更一般的屋顶R，这些屋顶满足以下性质：每个面都位于一个包含P的边的平面上，并且斜率为1。问题是，这样的R是否等于T(P)。结果表明，情况并非总是如此。然而，如果R的所有谷都与P相连，或者，如果对于任何点x ∈ R，最陡下降的路径通向P，则R和T(P)相等。换句话说，Aichholzer等人[AAAG95]表明，在所有屋顶中，T(P)具有特殊的性质，即下雨时它不会积水。

Following the notation of Cheng and Vigneron [CV07], we denote by aˆ the edge of T (G) that corresponds to the arc a in S (G). Analogously, we denote by fˆ(e) the facet of T (G) which corresponds to the face f (e) of S (G).

沿用 Cheng 和 Vigneron [CV07] 的符号，我们用 aˆ 表示 T (G) 的边，它对应于 S (G) 中的弧 a。类似地，我们用 fˆ(e) 表示 T (G) 的面，它对应于 S (G) 的面 f (e)。

1.2.4 The motorcycle graph

1.2.4 摩托车图

A straight-forward approach to computing the straight skeleton is through the simulation of the propagating wavefront. While edge events can be handled in a relatively efficient way the opposite holds for split events, see Section 1.4.2.1. It turns out to be non-trivial to efficiently determine which reflex wavefront vertex crashes into which wavefront edge. Let us consider for a moment only the simultaneous movement of the reflex wavefront vertices. In order to compute the straight skeleton it is important to know which reflex wavefront vertex is cutting off the trajectory of an other reflex vertex by reaching the common crossing point of their trajectories earlier. In order to extract this sub problem of computing straight skeletons, Eppstein and Erickson [EE99] introduced the so-called motorcycle graph.

计算直线骨架的一个直接方法是通过模拟传播波前。虽然边事件可以以相对有效的方式处理，但分割事件则相反，参见第1.4.2.1节。事实证明，有效确定哪个凹波前顶点撞击哪个波前边并非易事。让我们暂时只考虑凹波前顶点的同时移动。为了计算直线骨架，重要的是要知道哪个凹波前顶点通过更早地到达其轨迹的公共交叉点来切断另一个凹顶点的轨迹。为了提取计算直线骨架的这个子问题，Eppstein和Erickson [EE99] 引入了所谓的摩托车图。

A motorcycle is a point moving with constant speed on a straight line. Let us consider n motorcycles m1, . . . , mn, where each motorcycle mi has its own constant velocity vi ∈ R2 and a start point pi ∈ R2, with 1 ≤ i ≤ n. The trajectory {pi + t · vi : t ≥ 0} is called the track of mi. While a motorcycle moves it leaves a trace behind. When a motorcycle reaches the trace of another motorcycle then it stops driving — it crashes —, but its trace remains.

Note that it is possible that motorcycles never crash. Following the terminology of Eppstein and Erickson [EE99] such motorcycles are said to have escaped. It is easy to see that there can be up to (n2) intersections among the motorcycle tracks. However, no two motorcycle traces intersect in both interiors, which leads to at most n intersections among the traces.

摩托车是在直线上以恒定速度移动的点。让我们考虑n辆摩托车m1, . . . , mn，其中每辆摩托车mi都有其自身的恒定速度vi ∈ R2和一个起始点pi ∈ R2，其中1 ≤ i ≤ n。轨迹{pi + t · vi : t ≥ 0}被称为mi的轨道。当摩托车移动时，它会在身后留下痕迹。当一辆摩托车到达另一辆摩托车的轨迹时，它会停止行驶——它会撞车——但它的轨迹仍然存在。请注意，摩托车可能永远不会发生碰撞。按照 Eppstein 和 Erickson [EE99] 的术语，这样的摩托车被称为已经逃逸。很容易看出，摩托车轨迹之间最多可能有 (n2) 个交点。然而，没有两条摩托车轨迹在两个内部相交，这导致轨迹之间最多有 n 个交点。

Definition 1.8 (motorcycle graph).) The motorcycle graph M(m1, . . . , mn) of the motorcycles m1, . . . , mn is defined by the arrangement of the motorcycles traces.

定义 1.8 (摩托车图)[NT0]。) 摩托车 m1, . . . , mn 的 motorcycle graph M(m1, . . . , mn) 定义为摩托车轨迹的排列。

In Figure 5 (a) we illustrate the motorcycle graph M(m1, . . . , m10) of ten motorcycles. Cheng and Vigneron [CV07] presented a straight-skeleton algorithm for simple polygons P, which is based on the motorcycle graph, see Section 1.4.2.4. They first define a motorcycle graph induced by a polygon. The idea is that every reflex vertex v of P defines a motorcycle which starts at v and has the same velocity as the wavefront vertex which corresponds to v.

在图 5 (a) 中，我们展示了由十辆摩托车构成的摩托车图 M(m1, . . . , m10)。Cheng 和 Vigneron [CV07] 提出了一个针对简单多边形 P 的直线骨架算法，该算法基于摩托车图，见第 1.4.2.4 节。他们首先定义了一个由多边形诱导的摩托车图。其思想是，P 的每个凹顶点 v 定义了一辆摩托车，该摩托车从 v 出发，并具有与对应于 v 的波前顶点相同的速度。non-degeneracy assumption Cheng and Vigneron [CV07] explicitly exclude the case that vertex events appear for the wavefront of P. The authors call this the non-degeneracy assumption. Eppstein and Erickson [EE99] remarked that perturbation techniques cannot be applied to transform these “degeneracies” to general cases. A small perturbation would change the straight skeleton drastically, see Section 2.5.3. On the other hand, such situations are very likely to occur, in particular if P contains collinear edges. Figure 1 shows a typical example. (We will be present a generalization of the motorcycle graph to

arbitrary planar straight-line graphs in Section 2.5.) For the matter of simplicity we refer by the nondegeneracy assumption to the slightly more general assumption that no two motorcycles crash simultaneously into each other.[14]

非退化假设：Cheng和Vigneron [CV07] 明确排除了顶点事件出现在P波前的情况。作者称之为非退化假设。Eppstein和Erickson [EE99] 评论说，扰动技术不能用于将这些“退化”转换为一般情况。一个小的扰动会极大地改变直线骨架，参见第2.5.3节。另一方面，这种情况非常可能发生，特别是当P包含共线边时。图1 显示了一个典型的例子。（我们将在第2.5节中介绍摩托车图到任意平面直线图的推广。）为了简单起见，我们用非退化假设来指代稍微更一般的假设，即没有两辆摩托车同时相互碰撞。[14]

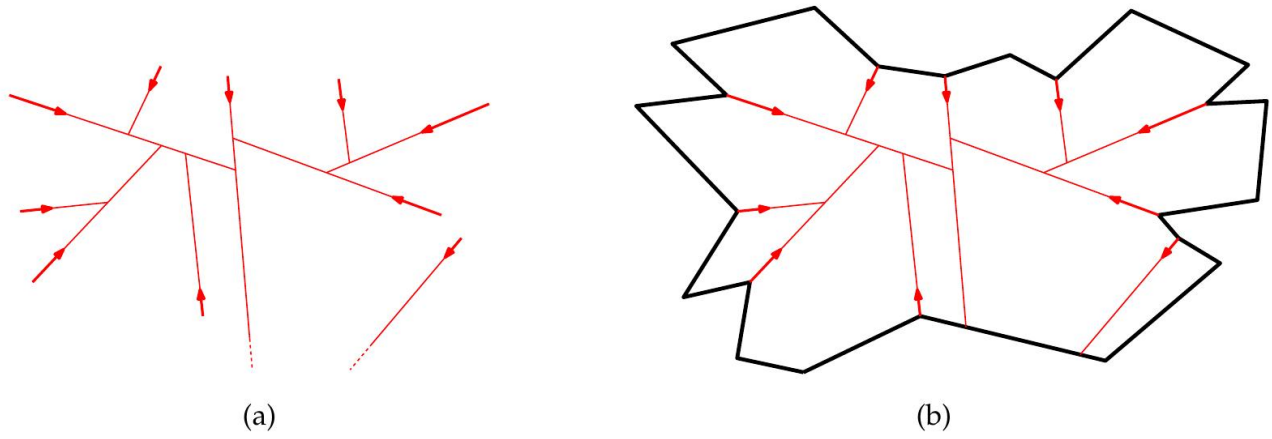


Figure 5: Left: The motorcycle graph M(m1, . . . , m10) of ten motorcycles. The velocities are represented by red arrows. Right: The motorcycle graph M(P), shown in red, induced by a simple polygon P (bold). Each reflex vertex of P emanates a motorcycle.

图 5：左图：十辆摩托车的摩托车图 M(m1, . . . , m10)。速度用红色箭头表示。右图：由简单多边形 P（粗体）导出的摩托车图 M(P)，以红色显示。 P 的每个凹顶点都发出一辆摩托车。

In order to guarantee the correctness of the algorithm of Cheng and Vigneron [CV07], it is necessary to assume that the motorcycles run out of fuel when they reach an edge of P. We cover this circumstance by introducing the alternative concept of walls. We assume that the plane contains straight-line segments which model rigid walls. If a motorcycle reaches a wall then it crashes and its trace remains. Following our terminology, we define a motorcycle graph induced by a polygon by specifying the motorcycles and the walls.

为了保证 Cheng 和 Vigneron [CV07] 算法的正确性，有必要假设摩托车在到达 P 的边缘时耗尽燃料。我们通过引入墙壁的替代概念来涵盖这种情况。我们假设该平面包含模拟刚性墙壁的直线段。如果摩托车到达墙壁，则会坠毁并留下痕迹。按照我们的术语，我们通过指定摩托车和墙壁来定义由多边形引起的摩托车图。

Definition 1.9 (motorcycle graph induced by a simple polygon). Let P denote a simple nondegenerate polygon. Each reflex vertex v emanates a motorcycle with the start point v and the same velocity as the corresponding wavefront vertex of v. Further, we consider the edges of P as walls. We denote by M(P) the resulting motorcycle graph and call M(P) the motorcycle graph induced by P.

定义 1.9 (由简单多边形诱导的摩托车图)[NT0]。设 P 表示一个简单的非退化多边形。每个凹顶点 v 发出一个摩托车，其起始点为 v，速度与 v 对应的波前顶点相同。此外，我们将 P 的边视为墙壁。我们将得到的摩托车图表示为 M(P)，并称 M(P) 为由 P 诱导的摩托车图。
Definition 1.10 (arm of a motorcycle). Let m denote a motorcycle of M(P) that emanates from the vertex v. We call the two wavefront edges that are incident to the reflex wavefront vertex emanated from v, the arms of m. The one arm that is left of the track of m is called left arm of m and the other arm is called right arm of m.

定义 1.10（摩托车的臂）[NT0]。设 m 表示从顶点 v 发出的 M(P) 的一辆摩托车。我们将与从 v 发出的凹波前顶点相连的两个波前边，称为 m 的臂。位于 m 轨迹左侧的臂称为 m 的左臂，另一个臂称为 m 的右臂。

We illustrate the motorcycle graph M(P) of a sample polygon P in Figure 5 (b). Cheng and Vigneron [CV07] also presented an extension of their algorithm to polygons with holes.

我们在图5（b）中展示了一个样本多边形P的摩托车图M(P)。Cheng和Vigneron [CV07]也提出了他们算法的扩展，以适用于带孔的多边形。
The motorcycle graph has to be extended accordingly: One has to introduce additional motorcycles at the convex vertices of the holes — which emanate reflex wavefront vertices within P — and one has to add the edges of the holes as walls, too.

摩托车图必须进行相应的扩展：必须在孔的凸顶点处引入额外的摩托车——这些顶点会发出P内的反射波前顶点——并且还必须将孔的边缘添加为墙壁。

1.3 a p p l i c a t i o n s

1.3 应用

In the following section, we present several applications that appeared since the introduction of straight skeletons. The application of shape reconstruction by Oliva et al. [OPC96] even appeared at roughly the same time as straight skeletons and, in fact, the authors referred to the straight skeleton by a different name, namely angular bisector networks (ABN). In general, straight skeletons inherit many applications from Voronoi diagrams. Two typical applications — computing offset curves and terrain modeling — are immediately connected to the original definition of straight skeletons by Aichholzer et al. [AAAG95] and Aichholzer and Aurenhammer [AA96].

在接下来的章节中，我们将介绍自直骨架引入以来出现的几个应用。Oliva等人[OPC96]提出的形状重建应用几乎与直骨架同时出现，事实上，作者用另一个名称，即角平分线网络（ABN）来指代直骨架。一般来说，直骨架继承了Voronoi图的许多应用。两个典型的应用——计算偏移曲线和地形建模——与Aichholzer等人[AAAG95]以及Aichholzer和Aurenhammer [AA96]对直骨架的原始定义直接相关。

1.3.1 Mitered offset curves and NC-machining

1.3.1 斜接偏移曲线与数控加工

Computational geometry has numerous applications in NC-machining. Computing offset curves is certainly one of the most important operations. Besides NC-machining, offset curves have a lot of further applications, like insetting/outsetting15 paths in vector graphics editors and CAD software, computing tolerance domains around polygons or polygonal chains (e. g. for approximations within given bounds), computing curves parallel to a given curve (e. g. territorial domains on the sea defined by some fixed distance from a coastline), and so on.

计算几何在数控加工中有大量的应用。计算偏移曲线无疑是最重要的操作之一。除了数控加工，偏移曲线还有许多其他的应用，例如在矢量图形编辑器和CAD软件中内插/外插15路径，计算多边形或多边形链周围的公差域（例如，在给定范围内进行近似），计算平行于给定曲线的曲线（例如，由距海岸线一定距离定义的海上领土范围）等等。

First, we introduce some technical terms related to NC-machining. A workpiece that should be milled by an NC-machine is represented by a simple polygon P, possibly with holes. The tool in operation has the shape of a disk Dr with radius r and the origin as center. In the domain of NC-milling such a polygon P is often called pocket and the tool is called cutter. If we would move the center of the cutter along the boundary of P, we obviously remove too much material, see Figure 6 (a). The idea is to “shrink” the polygon P and then move the tool along the new boundary such that we obtain the desired workpiece, see Figure 6 (b). In mathematical terms, we want to determine a shape P′ such that the Minkowski sum P′ + Dr := {x + y : x ∈ P′, y ∈ Dr} equals P (except for certain portions, e. g., at the convex corners of P). The boundary of P′ is called an offset curve of P with offset distance r. Note that there is only one continuous offset curve in Figure 6 (b) for that particular offset radius. If the tool did not leave the gap empty at the bottom of P then the bottom vertex of the hole would be cut off in this example.

首先，我们介绍一些与数控加工相关的技术术语。一个应该由数控机床铣削的工件由一个简单的多边形P表示，可能带有孔。运行中的刀具具有一个半径为r，原点为中心的圆盘Dr形状。在数控铣削领域，这样的多边形P通常被称为型腔，而刀具被称为铣刀。如果我们沿着P的边界移动铣刀的中心，我们显然会移除过多的材料，参见图6（a）。其思想是“收缩”多边形P，然后沿着新的边界移动刀具，从而获得所需的工件，参见图6（b）。用数学术语来说，我们想要确定一个形状P′，使得闵可夫斯基和P′ + Dr := {x + y : x ∈ P′, y ∈ Dr}等于P（除了某些部分，例如，在P的凸角处）。P′的边界被称为P的偏移曲线，偏移距离为r。请注意，在图6（b）中，对于该特定偏移半径，只有一条连续的偏移曲线。如果刀具没有在P底部留下空隙，那么在这个例子中，孔的底部顶点将被切掉。

At least two main questions related to computational geometry arise from the task of pocket machining: (i) how does one compute offset curves and (ii) how does one compute tool paths in order to remove the material in the interior of P? Held [Hel91] elaborated the computational problems related to pocket machining. He employs the Voronoi diagram of

至少有两个与计算几何相关的主要问题源于型腔加工的任务：（i）如何计算偏移曲线，以及（ii）如何计算刀具路径，以便去除P内部的材料？Held [Hel91] 详细阐述了与型腔加工相关的计算问题。他采用了Voronoi图

1.3 applications 13 - 靠岸学术

P in order to compute offset curves, where the offset curves are defined by the Minkowski difference P − Dr. Once the Voronoi diagram is available, this technique leads to a remarkably simple, efficient and robust method to compute offset curves, see [Hel91, HLA94]. Held [Hel91] also presented algorithms for two basic strategies in order to compute proper tool paths: contour parallel tool paths and direction parallel tool paths. Computing sophisticated tool paths is still a vital research area in these days. For example, Held and Spielberger [HS09] recently presented an algorithm that produces spiral tool paths which are suitable for high-speed machining. Computing an offset curve belongs to the first steps in their algorithm.

为了计算偏移曲线，P，其中偏移曲线由闵可夫斯基差 P − Dr 定义。一旦 Voronoi 图可用，这种技术就能产生一种非常简单、高效且稳健的计算偏移曲线的方法，参见 [Hel91, HLA94]。Held [Hel91] 还提出了两种基本策略的算法，以便计算合适的刀具路径：轮廓平行刀具路径和方向平行刀具路径。计算复杂的刀具路径在当今仍然是一个重要的研究领域。例如，Held 和 Spielberger [HS09] 最近提出了一种算法，该算法生成适用于高速加工的螺旋刀具路径。计算偏移曲线是他们算法中的第一步。

Voronoi-based offset curves consist of straight-line segments and circular arcs at the reflex vertices of P, see Figure 7 (a). Park and Chung [PC03] pointed out that such offset curves are undesirable if a high machining precision is required. The problem is that while the tool moves around a reflex vertex on a circular arc, it is permanently in touch with the reflex vertex, see Figure 7 (a). Oscillations of the workpiece or the tool during this time period lead to erosions of reflex vertices. Hence, they propose the so-called mitered offset curves, which are based on the straight skeleton instead of the Voronoi diagram. The basic idea is that sharp vertices of the polygon P should remain sharp in the offset curves, see Figure 7 (b). (Park and Chung [PC03] also mention that the offset curve based on the straight skeleton leads to long tool paths at sharp reflex vertices of the input. Hence, at reflex vertices with an interior angle of at least 1.5π the offset curves are trimmed by an additional line segment in order to shorten the tool path.)

基于Voronoi的偏移曲线由直线段和P的凹顶点处的圆弧组成，见图7(a)。Park和Chung [PC03]指出，如果需要高加工精度，这种偏移曲线是不理想的。问题在于，当刀具绕凹顶点在圆弧上移动时，它始终与凹顶点接触，见图7(a)。在此期间，工件或刀具的振动会导致凹顶点的腐蚀。因此，他们提出了所谓的斜接偏移曲线，该曲线基于直线骨架而不是Voronoi图。基本思想是多边形P的尖锐顶点应在偏移曲线中保持尖锐，见图7(b)。（Park和Chung [PC03]还提到，基于直线骨架的偏移曲线会导致输入中尖锐凹顶点处的刀具路径过长。因此，在内角至少为1.5π的凹顶点处，偏移曲线会被额外的线段修剪，以缩短刀具路径。）

The Voronoi diagram of simple polygons with holes — or more generally, of a planar straight-line graph — can be computed reliably in practice and implementations with an expected O(n log n) runtime exist, e. g., the Voronoi package Vroni by Held [Hel01]. In fact, Vroni also solves standard tasks for pocket machining, like computing offset curves, finding the maximum inscribed circle, or determining the medial axis16. Recently, Held and Huber [HH09a] extended the algorithms and the implementation behind Vroni to circular arcs. A nice byproduct of this extension is that the offset curves of straight-line polygons that are produced by Vroni, can serve as input to Vroni again.[17] As mentioned in the introduction of this chapter, the situation for the straight skeleton is clearly different. The only implementation available is the code by Cacciola [Cac04] within the CGAL library [CGA]. It accepts simple polygons with holes as input and exhibits a close-to quadratic runtime performance and memory footprint.

简单多边形（带孔）或更广义的平面直线图的Voronoi图可以在实践中可靠地计算，并且存在预期运行时间为O(n log n)的实现，例如Held的Voronoi包Vroni [Hel01]。事实上，Vroni也解决了口袋加工的标准任务，如计算偏移曲线、寻找最大内切圆或确定中轴线16。最近，Held和Huber [HH09a]将Vroni背后的算法和实现扩展到圆弧。这个扩展的一个很好的副产品是，Vroni产生的直线多边形的偏移曲线可以再次作为Vroni的输入。[17] 正如本章引言中提到的，直线骨架的情况明显不同。唯一可用的实现是Cacciola [Cac04]在CGAL库[CGA]中的代码。它接受带孔的简单多边形作为输入，并表现出接近二次的运行时性能和内存占用。

Park and Chung [PC03] circumvented the lack of efficient straight-skeleton algorithms by computing the mitered offset curves directly. Note that the mitered offset curve with offset distance t is equal to W (P, t). If no split event occurred until time t then one could simply compute the Voronoi diagram of P. After that one replaces the circular arcs of the offset curve by straight-line caps in order to obtain the mitered offset, without running into selfintersections of the offset curves. However, self intersections occur if the offset distance t is large enough such that a split event occurred until time t. Park and Chung [PC03] presented a relatively complex algorithm based on the concept of the so-called pairwise interference detection. Their algorithm basically puts offset segments parallel to the input edges and removes invalid portions of the offset curves afterwards.

Park 和 Chung [PC03] 通过直接计算斜接偏移曲线，规避了缺乏高效的直骨骼算法的问题。请注意，偏移距离为 t 的斜接偏移曲线等于 W (P, t)。如果在时间 t 之前没有发生分割事件，那么可以直接计算 P 的 Voronoi 图。之后，为了获得斜接偏移，可以用直线帽替换偏移曲线的圆弧，而不会遇到偏移曲线的自相交。然而，如果偏移距离 t 足够大，以至于在时间 t 之前发生了分割事件，则会发生自相交。Park 和 Chung [PC03] 提出了一种相对复杂的算法，该算法基于所谓的成对干扰检测的概念。他们的算法基本上放置与输入边平行的偏移线段，然后移除偏移曲线的无效部分。

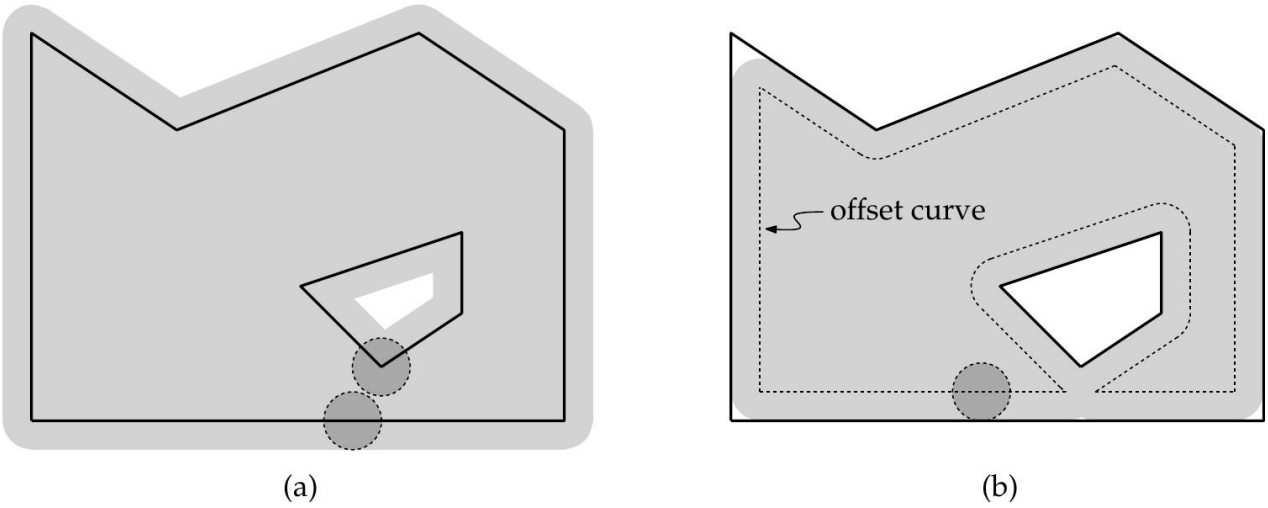


Figure 6: Milling a simple polygon P with a hole (bold). Left: Simply tracing the boundary of P with a tool removes too much material (shaded). Right: Tracing the offset curve (dashed) removes the correct amount of material, except for some remaining material at convex corners.

图 6：铣削一个带孔的简单多边形 P（粗体）。左图：简单地用刀具追踪 P 的边界会移除过多的材料（阴影部分）。右图：追踪偏移曲线（虚线）会移除正确数量的材料，但凸角处会残留一些材料。

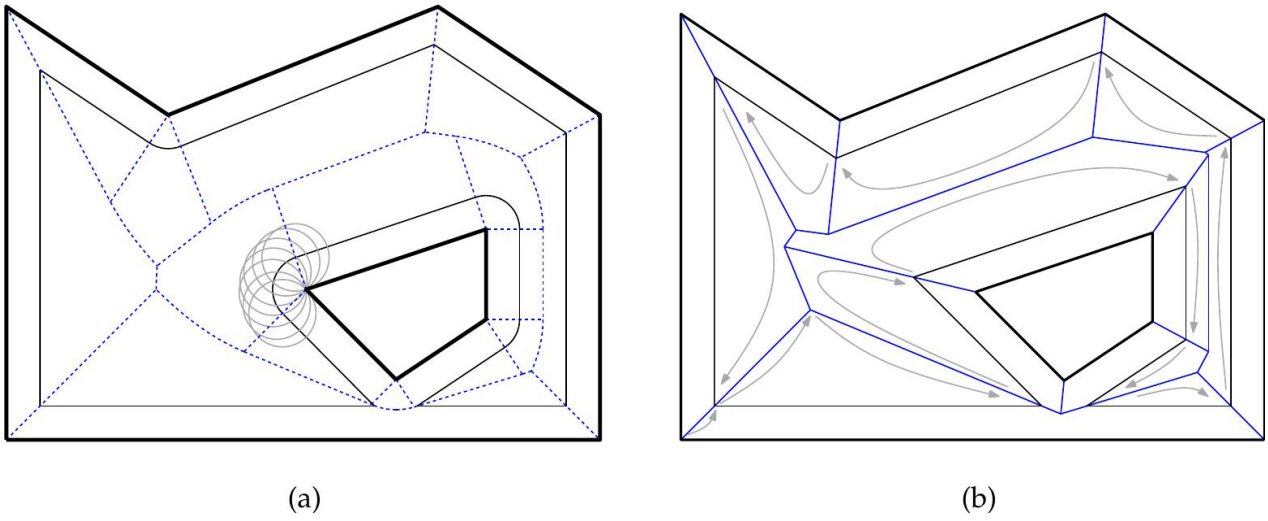


Figure 7: Two different approaches to obtain offset curves (thin) of the input (bold). Left: The approach based on the Voronoi diagram (dashed). The tool is shown in gray for several positions around are reflex vertex. Note that the tool remains in contact with the reflex vertex. Right: The approach based on the straight skeleton (dashed). The order, in which we traverse the straight skeleton to compute the offset curve, is sketched in gray.

图 7：获得输入曲线（粗体）的偏移曲线（细线）的两种不同方法。左图：基于 Voronoi 图（虚线）的方法。工具以灰色显示在反射顶点周围的几个位置。请注意，工具始终与反射顶点保持接触。右图：基于直骨架（虚线）的方法。我们遍历直骨架以计算偏移曲线的顺序以灰色草绘。

Obviously, if the straight skeleton is available, the mitered offset curves are computed in an almost trivial way by simply traversing the straight skeleton in the very same fashion as it is done for Voronoi-based offset curves. In Figure 7 (b) we sketch the order in which the straight skeleton is traversed for a particular offset distance. After we start from an arbitrary input vertex, we basically follow the boundary of one incident straight-skeleton face after the other and stop at each time when we reach the desired offset distance on a straight-skeleton arc.

显然，如果直线骨架可用，则斜接偏移曲线的计算几乎以一种非常简单的方式进行，只需以与基于Voronoi的偏移曲线相同的方式遍历直线骨架即可。在图7 (b) 中，我们勾勒出对于特定偏移距离遍历直线骨架的顺序。在我们从任意输入顶点开始后，我们基本上依次沿着一个入射直线骨架面的边界前进，并在每次到达直线骨架弧上所需的偏移距离时停止。

1.3.2 Building roofs and generating terrains

1.3.2 建造屋顶和生成地形

Automatic roof generation is an important task in 3D modeling, e. g., as part of an automatic city generator in a 3D modeling software. Let us consider a simple polygon P that represents the footprint of a building. How can we generate a realistic roof on top of the outer walls? A simple method to obtain a proper roof is to compute the roof model T (P) based on straight skeletons, see Section 1.2.3.

自动屋顶生成是 3D 建模中的一项重要任务，例如，作为 3D 建模软件中自动城市生成器的一部分。让我们考虑一个简单的多边形 P，它代表建筑物的占地面积。我们如何在外部墙壁的顶部生成一个逼真的屋顶？获得合适屋顶的一个简单方法是计算基于直骨架的屋顶模型 T (P)，参见第 1.2.3 节。

Laycock and Day [LD03] picked up this method and developed heuristic approaches in order to generate a larger variety of differently looking roofs. The original approach by interpreting the terrain T (P) above a simple polygon P as a roof results in the so-called hip roof. See Figure 8 for an example. Laycock and Day also describe how they obtain so-called gable roofs, mansard roofs, gambrel roofs, and Dutch roofs. For example, in order to generate a mansard roof, they consider the offset t until the first edge or split event happens and determine the offset at $0.85 \cdot t$, which is W (P, $0.85 \cdot t$). The mansard roof consists of the original facets of T (P) which are restricted up to an height of $0.85 \cdot t$ and a top facet parallel to the ground plane. Straight skeletons became a general tool in order to generate sophisticated and realistically looking roofs, see [MWH+06, KW11, Hav05] for further examples.

Laycock和Day [LD03] 采用了这种方法，并开发了启发式方法，以便生成更多样化的不同外观的屋顶。通过将简单多边形P上方的地形T (P)解释为屋顶，得到的原始方法产生了所谓的坡屋顶。参见图8中的示例。Laycock和Day还描述了他们如何获得所谓的山墙屋顶、孟莎屋顶、折线屋顶和荷兰屋顶。例如，为了生成孟莎屋顶，他们考虑偏移量t，直到发生第一个边或分割事件，并确定$0.85 \cdot t$处的偏移量，即W (P, $0.85 \cdot t$)。孟莎屋顶由T (P)的原始面组成，这些面被限制在高达$0.85 \cdot t$的高度，以及一个平行于地面的顶面。直骨架成为生成复杂且外观逼真的屋顶的通用工具，更多示例请参见[MWH+06, KW11, Hav05]。

The automatic generation of mountain terrains in the neighborhood of waters is very similar to roof construction. Assume we are given the shape of a river or a lake and we want to model the surrounding terrain. If the boundary of this shape is given by a planar straightline graph G then the terrain T (G) gives a realistic model for the surrounding terrain. In Figure 9 we show the generated terrain that illustrates the an part of the river Danube called “Schlögener Schlinge” in Austria. The generated terrain gives indeed a good approximation of the actual real-world scene.

在水域附近自动生成山地地形与屋顶建造非常相似。假设我们已知河流或湖泊的形状，并且我们想要模拟周围的地形。如果这个形状的边界由一个平面直线图G给出，那么地形T(G)为周围的地形提供了一个真实的模型。在图9中，我们展示了生成的地形，它展示了多瑙河的一部分，在奥地利被称为“Schlögener Schlinge”。生成的地形确实给出了实际真实世界场景的良好近似。

The 3D models of Figure 8 and Figure 9 were generated by our straight-skeleton implementation Bone, see Section 2.5.3. Our implementation can be used to export the terrain T (G) of a planar straight-line graph G in a file, which is further processed by the free 3D modeling software Blender [Ble].

图 8 和图 9 的 3D 模型由我们的直骨架实现 Bone 生成，参见第 2.5.3 节。我们的实现可用于在文件中导出平面直线图 G 的地形 T (G)，该文件由免费 3D 建模软件 Blender [Ble] 进一步处理。

1.3.3 Mathematical origami and the fold-and-cut problem

1.3.3 数学折纸与折叠切割问题

Straight skeletons possess an interesting application in the field of mathematical origami: the fold-and-cut problem. Let us consider a simple polygon P drawn on a sheet of paper. (The polygon P could illustrate a flower or an animal.) We are allowed to apply a sequence of folds of the paper along straight lines. Finally, we take a scissor and cut the folded paper along a straight line into pieces. Which sequence of folds and which line for the final cut

直骨架在数学折纸领域中有一个有趣的应用：折叠切割问题。让我们考虑一个绘制在纸上的简单多边形P。（多边形P可以表示一朵花或一只动物。）我们被允许沿着直线对纸张进行一系列的折叠。最后，我们用剪刀沿着一条直线剪开折叠的纸张，将其分成几块。哪种折叠顺序以及最终切割的哪条线

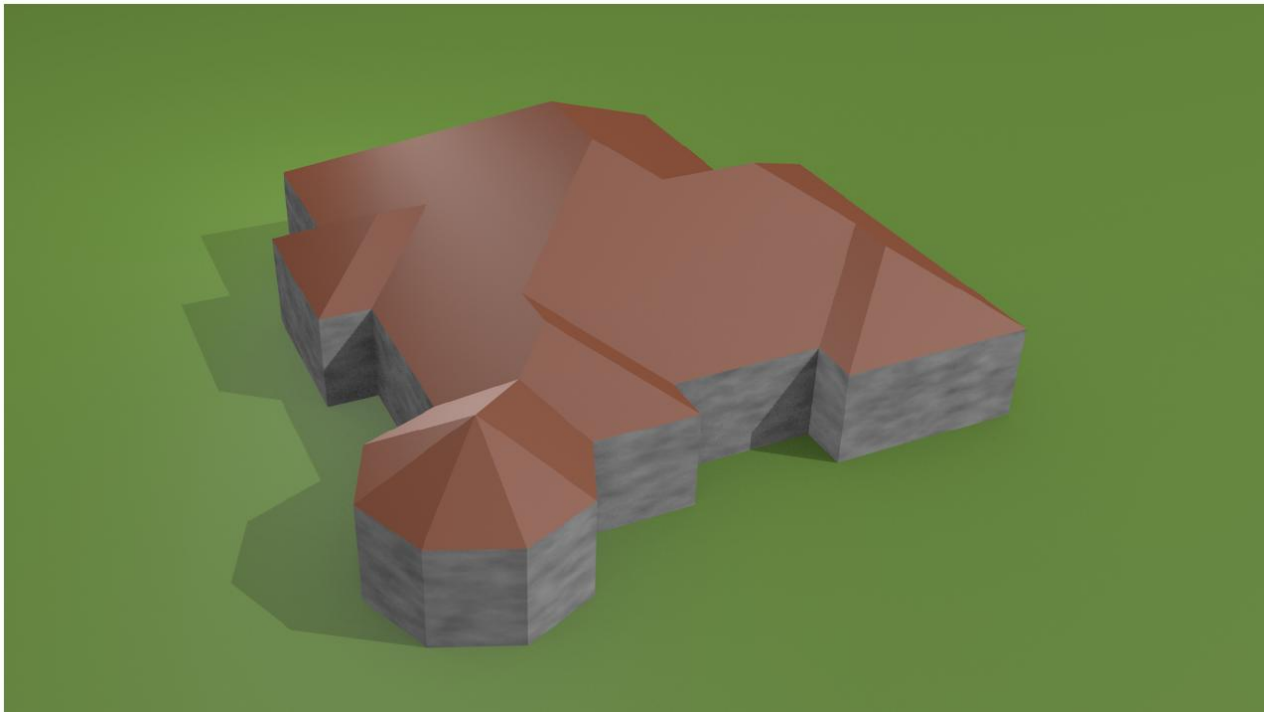


Figure 8: A hip roof generated by our straight-skeleton implementation Bone from the polygon that forms the footprint of the walls. Every facet of this roof has an identical slope.

图 8：由我们的直线骨架实现 Bone 从构成墙体轮廓的多边形生成的四坡屋顶。此屋顶的每个面都具有相同的坡度。

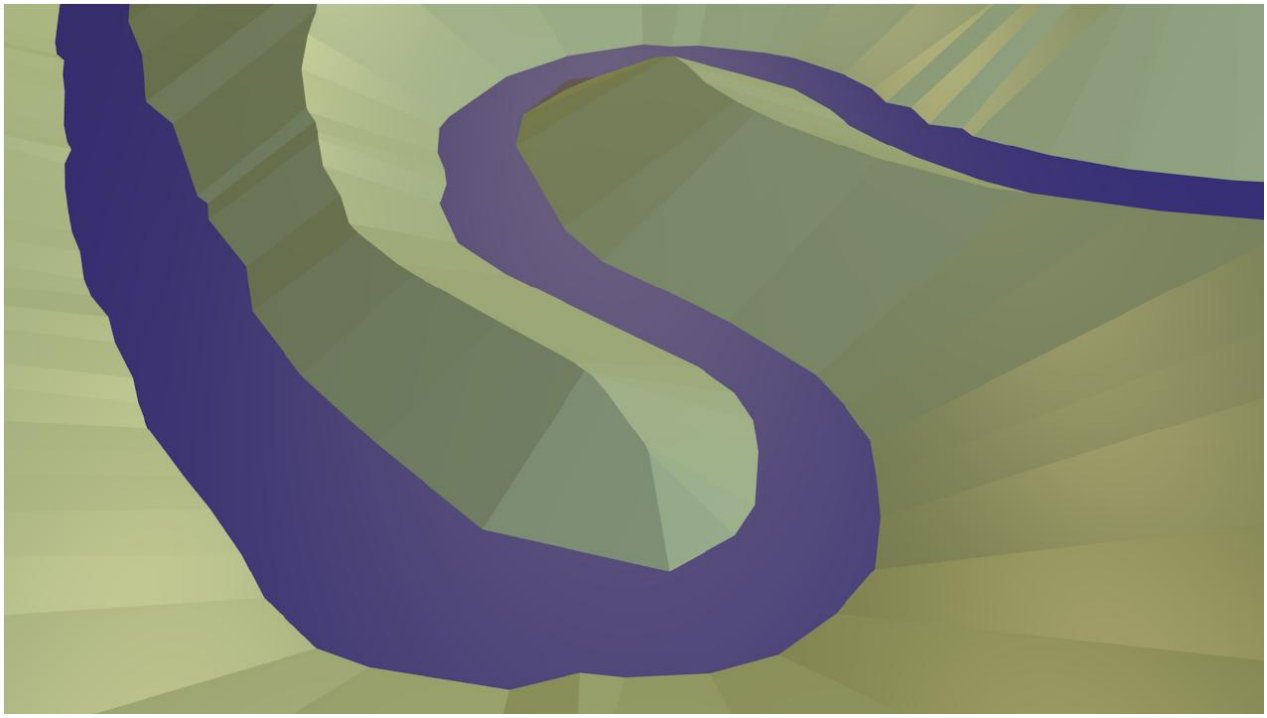


Figure 9: A terrain generated from the boundary of the blue river by our straight-skeleton implementation Bone. The figure illustrates the “Schlögener Schlinge"’, which is a part of the river Danube in Austria (48◦26’ 10” N 13◦51’ 50” E). The boundary of the river is based on data obtained from OpenStreetMap [OSM].

图 9：由我们的直线骨架实现 Bone 从蓝色河流边界生成的 terrain。该图展示了“Schlögener Schlinge”，它是奥地利多瑙河的一部分 (48◦26’ 10” N 13◦51’ 50” E)。河流的边界基于从 OpenStreetMap [OSM] 获取的数据。
do we have to apply in order to obtain a piece of paper that has the shape P? This problem, among others, is discussed in the book by Demaine and O’Rourke [DO07] in an illustrative and detailed fashion. They summarize the fold-and-cut problem as follows: (i) which shapes can be produced by such a fold-and-cut sequence and (ii) how can we compute a corresponding fold-and-cut sequence if the shape is given?

我们是否需要申请才能获得一张形状为P的纸？Demaine和O’Rourke在他们的著作[DO07]中以一种说明性和详细的方式讨论了这个问题以及其他问题。他们将折叠切割问题总结如下：（i）通过这种折叠切割序列可以产生哪些形状，以及（ii）如果给定形状，我们如何计算相应的折叠切割序列？

As described in [DO07], Chapter 17, the basic idea is to align the edges of P along straight lines by folding according to a so-called crease pattern. More precisely, the question whether a given polygon can be produced by a fold-and-cut sequence is equivalent to the question whether a crease pattern exists such that the edges of P can be arranged on a straight line and that any other point of the paper does not lie on this straight line. A cut through this line cuts exactly at the edges of P. The universality result states that every planar straight-line graph G can be cut by an appropriate fold-and-cut sequence [DDL98].

正如[DO07]第17章所述，基本思想是通过按照所谓的折痕图案进行折叠，使P的边沿直线对齐。更准确地说，给定的多边形是否可以通过折叠切割序列产生，等价于是否存在一种折痕图案，使得P的边可以排列在一条直线上，并且纸上的任何其他点都不位于这条直线上。沿着这条线切割正好切在P的边上。普遍性结果表明，每个平面直线图G都可以通过适当的折叠切割序列[DDL98]进行切割。

Demaine et al. [DDL98] presented an algorithm to compute crease patterns based on the straight skeleton. The input to their algorithm is a planar straight-line graph G. The motivation to use straight skeletons is that folding a paper at the bisector of two edges aligns both edges on a common straight line. Note that the arcs of the straight skeleton lie, by definition, on the bisectors of the defining pair of edges of G. The basic idea is that the straight skeleton almost poses an appropriate crease pattern (depending whether an arc is reflex or convex, the paper is folded in one or the other direction). As elaborated in [DO07], additional creases, so-called perpendicular creases, need to be introduced to obtain the final crease pattern for a given input graph G. Demaine et al. [DDL98] proved that a certain class of planar straight-line graphs can be obtained with a single cut using their crease-pattern algorithm based on straight skeletons. Crease patterns for arbitrary planar straight-line graphs can be computed using an algorithm based on disk packings. However, the crease patterns based on straight skeletons tend to be simpler [DO07].

Demaine等人[DDL98]提出了一种基于直线骨架计算折痕图案的算法。他们算法的输入是一个平面直线图G。使用直线骨架的动机是，在两条边的角平分线上折叠纸张会将两条边对齐在一条公共直线上。请注意，根据定义，直线骨架的弧线位于G的定义边对的角平分线

上。基本思想是，直线骨架几乎构成了一个合适的折痕图案（取决于弧线是凹的还是凸的，纸张会朝一个或另一个方向折叠）。正如[DO07]中详细阐述的那样，需要引入额外的折痕，即所谓的垂直折痕，以获得给定输入图G的最终折痕图案。Demaine等人[DDL98]证明，使用他们基于直线骨架的折痕图案算法，可以通过一次切割获得某一类平面直线图。可以使用基于圆盘填充的算法来计算任意平面直线图的折痕图案。然而，基于直线骨架的折痕图案往往更简单[DO07]。
Straight skeletons have also been applied to further problems in the field of mathematical origami and also to polyhedral wrapping problems, see [DDM00] for example.

直骨架也被应用于数学折纸领域的进一步问题，以及多面体包裹问题，例如参见[DDM00]。

1.3.4 Shape reconstruction and contour interpolation

1.3.4 形状重构和轮廓插值

Oliva et al. [OPC96] introduced an alternative version of the medial axis, which they called angular bisector network (ABN) and which turns out to be exactly the straight skeleton. Their motivation for this skeleton structure originates from the problem of 3D-surface reconstruction, for which they need to have a skeleton comprising straight-line segments only. They consider a sequence of cross-sections of a 3D-surface that lie on parallel planes. Each cross section consists of nested non-intersecting polygons defining so-called material and nonmaterial domains. The problem is to compute a reconstruction of the original 3D-surface based on the set of cross sections. This is a typical problem in medical imaging.

Oliva 等人 [OPC96] 引入了中轴线的一种替代版本，他们称之为角平分线网络 (ABN)，结果证明它恰好是直线骨架。他们提出这种骨架结构的动机源于 3D 表面重建问题，为此他们需要一个仅包含直线段的骨架。他们考虑位于平行平面上的 3D 表面的一系列横截面。每个横截面由定义所谓的材料和非材料域的嵌套的非相交多边形组成。问题是基于横截面集合计算原始 3D 表面的重建。这是医学成像中的一个典型问题。
The problem mentioned above is considered to be difficult in the presence of complex branches of the surface. Oliva et al. [OPC96] interpret the problem as an interpolation task between two consecutive sections. In the first step the polygonal shapes on two consecutive sections are projected on a parallel plane. Then they consider the symmetric difference of both shapes. This difference consists of nested polygons and each polygon comprises edges of both cross sections. In the next step they compute the straight skeleton of the difference shape and build a triangulation of the straight-skeleton faces. This triangulation is lifted to 3D in order to pose a patch of the 3D-surface between the two cross sections. The nodes of the straight skeleton lie on an intermediate layer between the two cross sections.

当表面存在复杂分支时，上述问题被认为是困难的。Oliva 等人 [OPC96] 将该问题解释为两个连续截面之间的插值任务。第一步，将两个连续截面上的多边形形状投影到平行平面上。然后，他们考虑两个形状的对称差。这种差异由嵌套多边形组成，每个多边形都包含

两个横截面的边。下一步，他们计算差异形状的直线骨架，并构建直线骨架面的三角剖分。为了在两个横截面之间放置 3D 表面的一个补丁，该三角剖分被提升到 3D。直线骨架的节点位于两个横截面之间的中间层上。
Barequet et al. [BGLSS04] presented a very similar approach. In contrast to [OPC96], they consider a different triangulation scheme and assign different heights to the vertices between the cross sections.

Barequet等人[BGLSS04]提出了一个非常相似的方法。与[OPC96]相反，他们考虑了一种不同的三角剖分方案，并为横截面之间的顶点分配了不同的高度。

1.3.5 Polygon decomposition

1.3.5 多边形分解

Tănase and Veltkamp [TV04b] proposed an approach to polygon decompositions based on straight skeletons, where the split events define how the polygon is decomposed into sub polygons. They consider a reflex wavefront vertex v that led to a split event by crashing into the wavefront edge e. Then the polygon is decomposed by (i) the arc a that is traced out by v and (ii) the projection line of the endpoint of a that is orthogonal to e, until the boundary of f (e) is hit. These two paths can be interpreted in the terrain model as two possible paths of a raindrop that starts at the endpoint of the lifted arc aˆ, see Section 1.2.3. Decompositions of this type form the first phase of their algorithm. In the second phase, further local simplifications of the decomposition are applied.

Tănase和Veltkamp [TV04b] 提出了一种基于直线骨架的多边形分解方法，其中分割事件定义了如何将多边形分解为子多边形。他们考虑一个反射波前顶点 v，该顶点通过撞击波前边缘 e 导致了分割事件。然后，多边形通过以下方式分解：（i）由 v 描绘的弧 a 和（ii）a 端点的投影线，该线垂直于 e，直到到达 f (e) 的边界。在地形模型中，这两条路径可以解释为从提升弧 aˆ 的端点开始的雨滴的两条可能路径，参见第 1.2.3 节。这种类型的分解构成了他们算法的第一阶段。在第二阶段，应用对分解的进一步局部简化。

The decomposition steps of the first phase are similar to the randomized partitioning used by Cheng and Vigneron [CV07]. However, Cheng and Vigneron [CV07] do not investigate polygon decomposition schemes per se, but used their partition in order to devise a randomized straight-skeleton algorithm, see Section 1.4.2.4.

第一阶段的分解步骤类似于 Cheng 和 Vigneron [CV07] 使用的随机分区。然而，Cheng 和 Vigneron [CV07] 本身并没有研究多边形分解方案，而是使用他们的分区来设计一种随机直线骨架算法，参见第 1.4.2.4 节。

1.3.6 Area collapsing in geographic maps and centerlines of roads

1.3.6 地图区域坍塌与道路中心线

In geographic information systems a common task is the simplification of maps by collapsing certain areas. Assume we are given a map of a landscape, including rivers, streets and municipalities. The rivers and streets are given as polygonal areas. The

problem of area collapsing asks for a method to collapse a certain polygonal area A (e. g., a river or a street) to a one-dimensional structure by dividing A among its neighboring areas.

在地理信息系统中，一项常见的任务是通过合并某些区域来简化地图。假设我们得到了一张景观地图，包括河流、街道和行政区域。河流和街道以多边形区域的形式给出。区域合并问题要求提供一种方法，通过在其相邻区域之间划分A，将某个多边形区域A（例如，河流或街道）合并为一维结构。
Haunert and Sester [HS08] introduced a method based on the straight skeleton S (A) of the polygon A. Each straight-skeleton face f (e) is merged with the neighboring area that shares the edge e with A. In order to assign larger portions of A to larger neighboring areas, they employ the weighted straight skeleton, cf. Section 1.5.2. The weighted straight skeleton allows them to adjust the sizes of the straight-skeleton faces in A accordingly. Their method respects certain topological constraints, e. g., it maintains connectivity. For instance, if a river joins a lake then it is desirable that the collapsed river is still connected to the lake.

Haunert和Sester [HS08] 引入了一种基于多边形 A 的直线骨架 S (A) 的方法。每个直线骨架面 f (e) 与共享边 e 的相邻区域和 A 合并。为了将 A 的较大部分分配给较大的相邻区域，他们采用了加权直线骨架，参见第 1.5.2 节。加权直线骨架允许他们相应地调整 A 中直线骨架面的大小。他们的方法尊重某些拓扑约束，例如，它保持连通性。例如，如果一条河流汇入一个湖泊，那么希望坍塌的河流仍然与湖泊相连。
Haunert and Sester [HS08] also discuss the problem of computing the centerlines of roads. They consider a road network of a city, where each road is given by the polygonal area it occupies. How can one obtain a one-dimensional representation such that each road is represented by a polygonal chain? Haunert and Sester [HS08] compute the so-called centerline of a road by, roughly speaking, considering those straight-skeleton arcs which are not defined by two adjacent boundary edges of the road area. Special care is taken in the presence of road junctions. If three or more roads meet in a junction it is rather unlikely that the four centerlines meet in a common straight-skeleton node. Haunert and Sester [HS08] presented a heuristic approach to detect such junctions and to connect the centerlines of the joining roads with a common junction node.

Haunert和Sester [HS08] 也讨论了计算道路中心线的问题。他们考虑一个城市的道路网络，其中每条道路都由其所占据的多边形区域给出。如何获得一个一维表示，使得每条道路都由一条折线链表示？Haunert和Sester [HS08] 通过粗略地考虑那些不是由道路区域的两个相邻边界边定义的直骨架弧来计算道路的所谓中心线。在存在道路交叉口的情况下，需要特别注意。如果三条或更多道路在一个交叉口相遇，那么四条中心线不太可能在一个共同的直骨架节点相遇。Haunert和Sester [HS08] 提出了一种启发式方法来检测此类交叉口，并将连接道路的中心线与一个共同的交叉口节点连接起来。

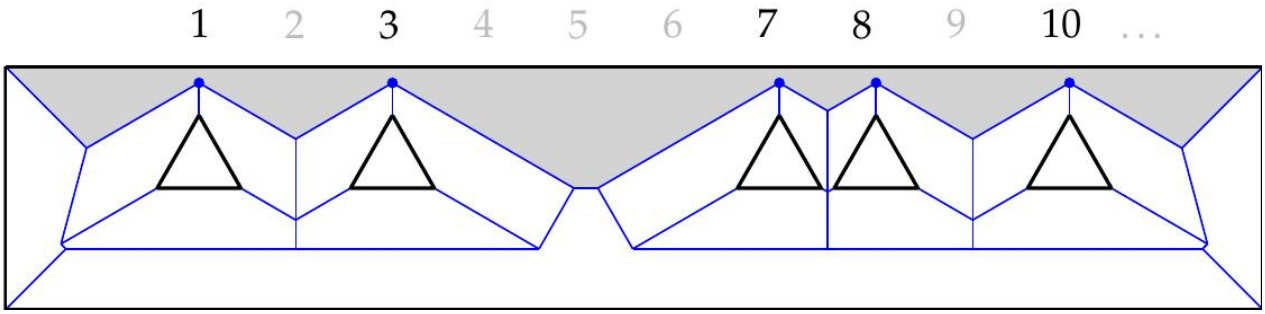


Figure 10: Sorting natural numbers can be reduced to the computation of the straight skeletons (blue) of a polygon with holes (bold). The nodes that are depicted by small disks correspond to the input numbers {1, 3, 7, 8, 10}.

图10：自然数的排序可以简化为带孔多边形（粗体）的直线骨架（蓝色）的计算。由小圆盘表示的节点对应于输入数字{1, 3, 7, 8, 10}。

1.4 p r i o r w o r k

1.4 前人工作

1.4.1 Runtime bounds for the straight skeleton

1.4.1 直骨架的运行时间界限

To the best of our knowledge, the currently best known lower bound to compute the straight skeleton of a simple polygon with n vertices is Ω(n). In the convex case a simulation of the propagating wavefront gives us an almost optimal algorithm using O(n log n) time, see Section 1.4.2.1. For the more general case of monotone polygons, Das et al. [DMN+10] presented an O(n log n) algorithm. However, for arbitrary simple polygons, no algorithm that is even reasonably close to linear is known. Currently, the fastest algorithm is due to Eppstein and Erickson [EE99] with a worst-case runtime of O(n17/11+ϵ). This poses a high contrast to the situation for Voronoi diagrams, for which Chin and Snoeyink [CSW99] presented an optimal linear-time algorithm. Note that in the convex case the Voronoi diagram and the straight skeleton are coinciding and the algorithm by Chin and Snoeyink solves the straight-skeleton problem in optimal time as well. However, for convex polygons also a simpler Voronoi algorithm is known due to Aggarwal et al. [AGSS87].

据我们所知，目前计算具有n个顶点的简单多边形的直线骨架的最佳已知下界是Ω(n)。在凸情况下，传播波前的模拟为我们提供了一个几乎最优的算法，使用O(n log n)时间，参见第1.4.2.1节。对于更一般的单调多边形情况，Das等人[DMN+10]提出了一个O(n log n)算法。然而，对于任意简单多边形，目前还没有已知算法能合理地接近线性时间复杂度。目前，最快的算法是Eppstein和Erickson [EE99]提出的，其最坏情况运行时间为O(n17/11+ϵ)。这与Voronoi图的情况形成了鲜明对比，Chin和Snoeyink [CSW99]为此提出了一个最优的线性时间算法。请注意，在凸情况下，Voronoi图和直线骨架是重合的，Chin和Snoeyink的算法也以最佳时间解决了直线骨架问题。然而，对于凸多边形，Aggarwal等人[AGSS87]也提出了一个更简单的Voronoi算法。

For planar straight-line graphs and for polygons with holes, the best known lower bound is Ω(n log n). This is easy to see, since we can simply reduce the sorting problem to straight skeletons. Assume n distinct natural numbers a1, . . . , an are given, which have to be sorted. For any ak, we place a small equilateral triangle that is hinged with its top vertex at the coordinates (ak, 0). The length of the edges of the triangles are considered to be less than 1, say 0.9. Then we put a box around the triangles such that the top vertices of the box have small y-coordinates, say 0.1. The box and the triangles form a simple polygon P with holes with 3n + 4 vertices and can be constructed in O(n) time. After computing S (P) we consider the face f (e) of the top edge e of P that is within P. It is easy to see that the nodes that appear as reflex vertices of f (e) correspond to the input a1, . . . , an and occur in sorted order along the x-axis, see Figure 10. A simple traverse of f (e) gives us the sorted sequence in linear time.

对于平面直线图和带孔多边形，目前已知的最佳下界是Ω(n log n)。这很容易理解，因为我们可以简单地将排序问题归约为直线骨架。假设给定n个不同的自然数a1, . . . , an，需要对它们进行排序。对于任意ak，我们放置一个小等边三角形，其顶部顶点铰接在坐标(ak, 0)处。三角形的边长被认为小于1，比如0.9。然后我们在三角形周围放置一个盒子，使得盒子的顶部顶点具有较小的y-坐标，比如0.1。盒子和三角形形成一个具有3n + 4个顶点且带孔的简单多边形P，并且可以在O(n)时间内构造。在计算S (P)之后，我们考虑P的顶部边e的面f (e)，该面位于P内。很容易看出，作为f (e)的凹顶点出现的节点对应于输入的a1, . . . , an，并且沿x-轴按排序顺序出现，参见图10。简单地遍历f (e)就可以在线性时间内得到排序后的序列。

1.4.2 Algorithms for computing straight skeletons and motorcycle graphs

1.4.2 计算直线骨架和摩托车图的算法

1.4.2.1 Aichholzer et al., 1995

1.4.2.1 Aichholzer 等人，1995

When Aichholzer et al. [AAAG95] introduced straight skeletons of simple polygons P, they also presented an algorithm, which simulates the propagation of the wavefront in a discrete manner. Their algorithm maintains a priority queue Q which contains all potential edge events for each wavefront edge of P, prioritized by their occurrence time. Assume for a moment that P is convex, which means that only edge events occur. The algorithm fetches the earliest event from Q and processes it accordingly. That is, the incident vertices of the affected edge e are merged to a single vertex v. Note that the velocity of v is given by the two incident wavefront edges. Hence, the collapsing times of the two incident edges of v got invalid and one has to (i) re-compute them and (ii) update the priority queue accordingly. A single edge event affects only a constant number of entries within Q and can therefore be handled in O(log n) time. For convex polygons P this algorithm computes the straight skeleton in O(n log n) time.

当Aichholzer等人[AAAG95]引入简单多边形的直骨架P时，他们也提出了一种以离散方式模拟波前传播的算法。他们的算法维护一个优先级队列Q，其中包含P的每个波前边的所有潜在边事件，并按其发生时间排序。假设P是凸的，这意味着只发生边事件。该算法从Q中

获取最早的事件并相应地处理它。也就是说，受影响的边e的入射顶点被合并到单个顶点v。请注意，v的速度由两个入射波前边给出。因此，v的两个入射边的塌陷时间变得无效，必须（i）重新计算它们，并且（ii）相应地更新优先级队列。单个边事件仅影响Q中的常数个条目，因此可以在O(log n)时间内处理。对于凸多边形P，该算法在O(n log n)时间内计算直骨架。
In the presence of split events the situation gets more complicated. Efficiently determining the first split turns out to be non-trivial. Let v denote a reflex vertex of P. The problem is to determine in which wavefront edge the corresponding reflex wavefront vertex will crash. Simple ray-shooting does not solve the problem. The wavefront edge e, which is hit by the ray, could move off the ray during the wavefront propagation. Maintaining the wavefront edge that is hit by the ray does not solve the problem either: another reflex wavefront vertex from aside could cross the ray at any position. However, detecting such incidents seems to be costly.

在存在分裂事件的情况下，情况变得更加复杂。有效地确定第一次分裂变得非常重要。设v表示P的一个凹顶点。问题是确定相应的凹波前顶点将在哪个波前边上碰撞。简单的光线投射并不能解决问题。被光线击中的波前边e可能会在波前传播过程中偏离光线。保持被光线击中的波前边也无法解决问题：来自侧面的另一个凹波前顶点可能在任何位置穿过光线。然而，检测此类事件似乎代价高昂。

The idea of Aichholzer et al. [AAAG95] is the following: Consider two consecutive edge events at time t′ and $t > t'$ and assume that the wavefront is free of self-intersections until time t′. The wavefront becomes self-intersecting until t if and only if one or more split events happened in the time interval [t′, t]. More importantly, Aichholzer et al. [AAAG95] were able to show that these split events can be determined from the wavefront W (P, t). In particular, they showed that the corresponding split events can be found and processed in O(n log n) time.

Aichholzer等人[AAAG95]的思路如下：考虑时间t′和$t > t'$的两个连续边事件，并假设波前在时间t′之前没有自相交。当且仅当在时间区间[t′, t]内发生一个或多个分裂事件时，波前才会自相交直到t。更重要的是，Aichholzer等人[AAAG95]能够证明这些分裂事件可以从波前W (P, t)确定。特别地，他们证明了可以在O(n log n)时间内找到并处理相应的分裂事件。

Testing whether a self-intersection is present at the k-th edge event is done in linear time using the triangulation algorithm by Chazelle [Cha91]. Applying exponential searching, the first split event can be determined at the costs of O(log k) intersection tests, i. e., in O(n log k) time, where the first split event happened just before the k-th edge event. Applying this algorithm recursively on each sub-polygon of W (P, t) after the first split event leads to a total runtime of O(n2 log n). Denoting by $r \in O(n)$ the number of reflex vertices, one can further refine the runtime analysis to O(nr log n).

使用 Chazelle [Cha91] 的三角剖分算法，可以在线性时间内完成对第 k 个边事件处是否存在自相交的测试。应用指数搜索，确定第一个分裂事件的成本为 O(log k) 次相交测试，即 O(n log k) 时间，其中第一个分裂事件发生在第 k 个边事件之前。在第一次分裂事件之后，对 W (P, t) 的每个子多边形递归地应用此算法，将导致总运行时间为 O(n2 log n)。用 $r \in O(n)$ 表示凹顶点的数量，可以进一步将运行时间分析细化到 O(nr log n)。

1.4.2.2 Aichholzer and Aurenhammer, 1996

1.4.2.2 Aichholzer和Aurenhammer，1996

Aichholzer and Aurenhammer [AA96, AA98] presented an algorithm for planar straightline graphs which is based on a kinetic triangulation. Their algorithm accepts a planar straight-line graph G with n vertices as input. The basic idea is again to simulate the propagating wavefront, but to exploit the topological changes in a kinetic triangulation in order to determine the edge and split events.

Aichholzer和Aurenhammer [AA96, AA98] 提出了一种基于动态三角剖分的平面直线图算法。他们的算法接受一个具有n个顶点的平面直线图G作为输入。其基本思想仍然是模拟传播的波前，但利用动态三角剖分中的拓扑变化来确定边事件和分裂事件。

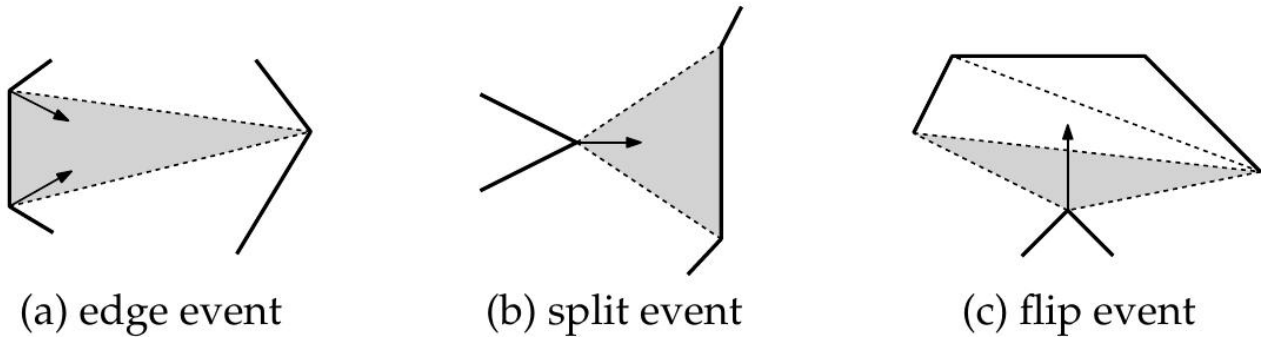


Figure 11: The three types of topological changes during the propagation of the triangulation.

图 11：三角剖分传播期间的三种拓扑变化。

The algorithm starts with an initial triangulation of G and keeps the area St′≥t W (G, t′) triangulated for all times t ≥ 0. Aichholzer and Aurenhammer [AA96] also include infinite triangles to the triangulation such that the entire plane is triangulated at time zero. During the propagation of the wavefront, the triangulation keeps the area St′≥t W (G, t′) triangulated and at certain points in time the triangulation may change its topology: triangles collapse to a point or to a line. Whenever such an event occurs local modifications have to be applied in order to maintain a proper triangulation. The essential observation is that edge and split events of the wavefront correspond to a collapse of a triangle. Unfortunately, not every triangle collapse corresponds to an event of the wavefront. Aichholzer and Aurenhammer [AA96] distinguish the following types of events, as illustrated in Figure 11.

该算法从G的初始三角剖分开始，并保持面积St′≥t W (G, t′)在所有时间 t ≥ 0 内都被三角剖分。Aichholzer 和 Aurenhammer [AA96] 还在三角剖分中包含了无限三角形，使得整个平面在时间零时被三角剖分。在波前传播过程中，三角剖分保持面积St′≥t W (G, t′)被三角剖分，并且在某些时间点，三角剖分可能会改变其拓扑结构：三角形坍缩为一个点或一条线。每当发生此类事件时，必须应用局部修改以保持适当的三角剖分。重要的观察结果是，波前的边事件和分裂事件对应于三角形的坍缩。不幸的是，并非每个三角形的坍缩都对应于波前的一个事件。Aichholzer 和 Aurenhammer [AA96] 区分了以下几种类型的事件，如图 11 所示。

• Edge event: A triangle collapsed due to an edge event of the wavefront. The corresponding edge e collapsed to zero length and the one triangle having e as an edge collapsed with it.

• 边事件：一个三角形由于波前的边事件而坍塌。相应的边e坍塌为零长度，并且以e为边的三角形随之坍塌。

• Split event: A triangle collapsed because a split event in the wavefront happened. In the simple case a reflex wavefront vertex moves into a wavefront edge e and the one triangle having e as an edge collapses thereby. In case of a multi split event two or more reflex wavefront vertices meet in a point and again certain triangles collapse.

• 分裂事件：由于波前中发生分裂事件，导致一个三角形坍塌。简单的情况是，一个反射波前顶点移动到一个波前边e中，并且以e为边的三角形会因此坍塌。在多重分裂事件的情况下，两个或多个反射波前顶点在一个点相遇，同样会导致某些三角形坍塌。

• Flip event: A triangle collapsed because a wavefront vertex crosses an inner triangulation diagonal. The affected diagonal has to be flipped18 in the triangulation in order to maintain a valid triangulation. This event does not immediately correspond to a topological change of the wavefront.

• 翻转事件：由于波前顶点穿过内部三角剖分对角线而导致三角形塌陷。为了保持有效的三角剖分，必须翻转18受影响的对角线。此事件并不立即对应于波前的拓扑变化。

The algorithm of Aichholzer and Aurenhammer computes S (G) by keeping track of the topological changes within the triangulation. This again involves a priority queue Q containing the events. The runtime complexity depends on the number of events that occurred. The number of edge and split events corresponds to the number of nodes of S (G). According to Lemma 2.4, this number[19] is in O(n). Each edge and split event involves the adaption of a certain number of wavefront vertices and, as a consequence, the re-calculation of the collapsing times of up to O(n) triangles that are incident to these vertices. Summarizing, all edge and split events can be handled in O(n2 log n) time.

Aichholzer和Aurenhammer的算法通过跟踪三角剖分中的拓扑变化来计算S (G)。这再次涉及包含事件的优先级队列Q。运行时间复杂度取决于发生的事件数量。边事件和分裂事件的数量对应于S (G)的节点数量。根据引理2.4，这个数字[19]是O(n)量级的。每个边事件和分裂事件都涉及一定数量的波前顶点的调整，并因此需要重新计算最多O(n)个与这些顶点关联的三角形的坍缩时间。总而言之，所有边事件和分裂事件都可以在O(n2 log n)时间内处理。

However, finding a sophisticated upper bound for the number of flip events appears to be harder. First of all, a single flip event is handled in O(log n) time since it only involves the two triangles that share the affected triangulation diagonal. The upper bound of O(n3) for the number of flip events is relatively easy to see. Each wavefront vertex moves along a straight-line with constant speed. Consider the vertices p, q, r of a triangle Δ and denote by p(t), q(t), r(t) their positions at time t, respectively. We denote by u, v, w $\in R^2$ the velocities of p, q, r. The triangle Δ collapses if and only if the points p, q, r get collinear. That is

然而，为翻转事件的数量找到一个更精细的上界似乎更难。首先，处理单个翻转事件的时间复杂度为O(log n)，因为它只涉及到共享受影响三角剖分对角线的两个三角形。翻转事件数量的上界O(n3)相对容易理解。每个波前顶点都以恒定速度沿直线移动。考虑三角形Δ的顶点p、q、r，并分别用p(t)、q(t)、r(t)表示它们在时间t的位置。我们用u、v、w ∈ R^2表示p、q、r的速度。三角形Δ塌陷当且仅当点p、q、r共线。也就是说

□

□

□

□

□

□

p

x

(

t

)

q

x

(

t

)

r

x

(

t
)
p
y
(
t
)
q
y
(
t
)
r
y
(
t
)
1
1
1
□
□
□

$$= p_x(0) + t \cdot u_x q$$

x
(
0
)
+
t
.
v
x
r
x
(
0
)
+
t
.
w
x
p
y
(
0
)
+
t

.

u

y

q

y

(

0

)

+

t

.

v

y

r

y

(

0

)

+

t

.

w

y

1

1

1

=

0,

(1.2)

where the x- and y-coordinates are denoted by subscriptions. This quadratic equation in t is fulfilled for exactly zero, one, two or all values of t. As a consequence, a single triangle with vertices p, q, r collapses at most twice. Since we have (n3) possible triples of vertices the number of flip events is bounded by O(n3). Summarizing, the runtime complexity of the algorithm is in O((n2 + k) log n), where k ∈ O(n3) denotes the number of flip events.

其中，x和y坐标用下标表示。关于t的这个二次方程对于恰好零个、一个、两个或所有t的值成立。因此，具有顶点p、q、r的单个三角形最多坍缩两次。由于我们有(n3)个可能的顶点三元组，翻转事件的数量受到O(n3)的限制。总而言之，该算法的时间复杂度为O((n2 + k) log n)，其中k ∈ O(n3)表示翻转事件的数量。

Interestingly, to the best of our knowledge, no input is known that exceeds a quadratic number of flip events. We will revisit the open question concerning this gap of a linear factor in Section 2.2. For some typical input data Aichholzer and Aurenhammer [AA96] observe an actual runtime close to O(n log n). However, no published runtime statistics are known.

有趣的是，据我们所知，目前还没有已知的输入能够超过二次数量的翻转事件。我们将在第2.2节重新讨论关于线性因子差距的未决问题。对于一些典型的输入数据，Aichholzer和Aurenhammer [AA96] 观察到实际运行时间接近O(n log n)。然而，目前还没有已发表的运行时间统计数据。

1.4.2.3 Eppstein and Erickson, 1999

1.4.2.3 Eppstein和Erickson，1999

Eppstein and Erickson [EE99] presented a straight-skeleton algorithm for simple polygons, which is applicable to planar straight-line graphs and which can even be extended to compute the weighted straight skeleton, cf. Section 1.5.2. Let us recall the discussion in Section 1.4.2.1 concerning the determination of the split events. As elaborated by Aichholzer et al. [AAAG95], a simple ray-shooting from reflex vertices does not solve the problem of finding the next split event. However, one could consider for every propagating wavefront edge e the triangle (possibly infinite in size) that is bound by e at time zero and the trajectories of its incident wavefront vertices. The idea is to keep track of those pairs of reflex vertices v and wavefront edges e, where the potential split event of v and e takes place within the corresponding triangle of e. The next split event is given by closest pair in terms of the earliest split-event time.

Eppstein和Erickson [EE99] 提出了一个针对简单多边形的直骨架算法，该算法适用于平面直线图，甚至可以扩展到计算加权直骨架，参见第1.5.2节。让我们回顾一下第1.4.2.1节中关于确定分裂事件的讨论。正如Aichholzer等人 [AAAG95] 详细阐述的那样，从凹顶点进行简单的射线投射并不能解决找到下一个分裂事件的问题。然而，可以考虑对于每个传播波前边缘 e，由时间零时的 e 及其入射波前顶点的轨迹所限定的三角形（可能大小无限）。其思想是跟踪那些凹顶点 v 和波前边缘 e 的对，其中 v 和 e 的潜在分裂事件发生在 e 的相应三角形内。下一个分裂事件由最早分裂事件时间方面的最近对给出。

Eppstein and Erickson [EE99] pursued this approach by employing very powerful closestpair data structures. First of all, Eppstein and Erickson lift the problem to R3 in the same fashion as it is done for the terrain model T (G). Every reflex vertex v of W (G, 0) defines a ray in R3 starting at v and supports the corresponding ridge in T (G). At every initial wavefront edge e a corresponding (lifted) triangle, which supports fˆ(e), is considered. The two other edges of the triangle are given by the supporting rays of the lifted trajectories of the incident wavefront vertices, see Figure 12. Note that if we consider the unweighted straight skeleton then the triangles have slope 1, whereas the rays have a slope of at most 1.

Eppstein和Erickson [EE99] 采用这种方法，使用了非常强大的最近点对数据结构。首先，Eppstein和Erickson将问题提升到R3，其方式与地形模型T (G)的处理方式相同。W (G, 0) 的每个凹顶点 v 在 R3 中定义一条以 v 为起点的射线，并支撑 T (G) 中相应的山脊。在每个初始波前边缘 e 处，考虑一个相应的（提升的）三角形，该三角形支撑 fˆ(e)。三角形的另外两条边由入射波前顶点的提升轨迹的支撑射线给出，参见图 12。请注意，如果我们考虑未加权的直线骨架，则三角形的斜率为 1，而射线的斜率最大为 1。

Eppstein and Erickson [EE99] consider a sweep plane, which sweeps R3 along the z-axis. Whenever the sweep plane reaches the top of a triangle, an edge event happens. When the intersection of a ray with a triangle is reached a split happened, including multi split

Eppstein和Erickson [EE99] 考虑一个扫描平面，该平面沿着z-轴扫描R3。每当扫描平面到达三角形的顶部时，就会发生边事件。当射线与三角形的交点到达时，会发生分割，包括多重分割

The triangle of an edge e and a ray of the reflex vertices v, as considered by the algorithm

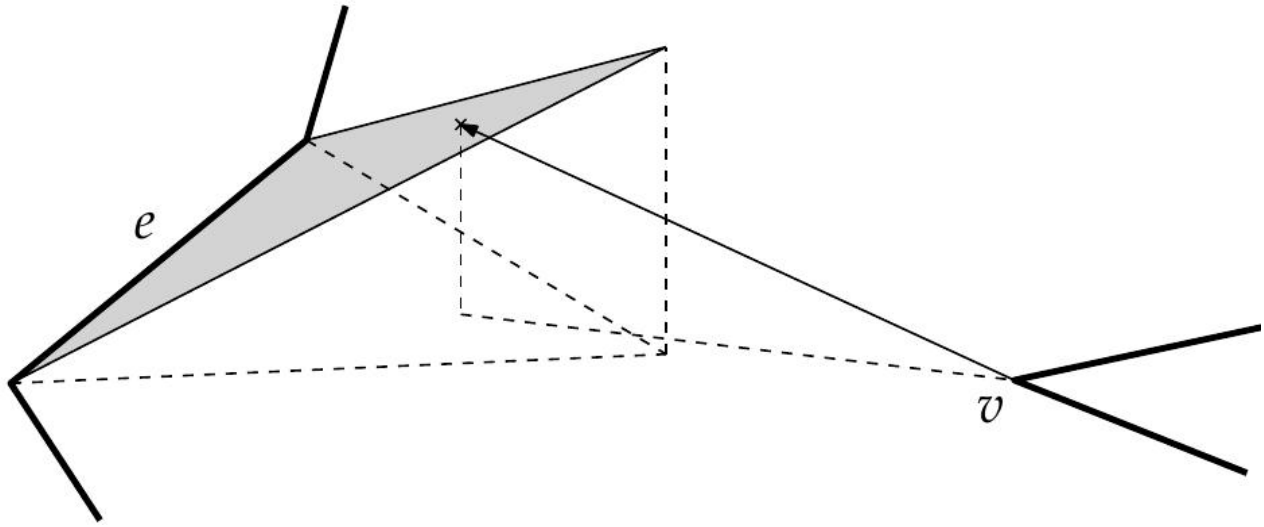


Figure 12: The triangle of an edge e and a ray of the reflex vertices v, as considered by the algorithm by Eppstein and Erickson [EE99]. The lowest intersection of a ray with a triangle gives the next split event.

图 12：边 e 的三角形和凹顶点 v 的射线，如 Eppstein 和 Erickson [EE99] 算法中所考虑的那样。射线与三角形的最低交点给出了下一个分割事件。

Central aspects in the algorithm by Eppstein and Erickson [EE99] are range searching techniques that allow the application of time-space trade-offs. Assume we are in possession of a data structure that supports fast queries but has a high space consumption and, conversely, a data structure that has high query times but low space bounds. The idea is to mix those two data structure in a hierarchical fashion in order to obtain characteristics for query time and space consumption between the two original data structures. A survey by Agarwal and Erickson [AE99] discusses these techniques in detail.

Eppstein和Erickson [EE99]算法的核心在于范围搜索技术，该技术允许时间-空间权衡的应用。假设我们拥有一个支持快速查询但空间消耗高的数据结构，反之，也拥有一个查询时间长但空间界限低的数据结构。其思想是以分层方式混合这两种数据结构，以便获得介于两种原始数据结构之间的查询时间和空间消耗特性。Agarwal和Erickson [AE99]的综述详细讨论了这些技术。

Eppstein and Erickson [EE99] denote by s the space required by a data structure and express the time complexity as a function in s. One step towards their straight-skeleton algorithm is a data structure which itself is based on several other data structures and allows lowest intersection queries among n rays and triangles within O(n1+ϵ/s1/4) time, for any ϵ > 0. Another main ingredient is a data structure by Agarwal and Matousekˇ

[AM94] in order to answer ray shooting queries, again in O(n1+ϵ/s1/4) time. By balancing time and space complexities, Eppstein and Erickson finally obtain a straight-skeleton algorithm which uses O(n8/5+ϵ) time and space. They further note that this algorithm is also applicable in order to compute the weighted straight skeleton.

Eppstein 和 Erickson [EE99] 用 s 表示数据结构所需的空间，并将时间复杂度表示为 s 的函数。他们的直线骨架算法的一个步骤是基于几个其他数据结构的数据结构，该数据结构允许在 O(n1+ϵ/s1/4) 时间内查询 n 条射线和三角形之间的最低交点，对于任何 ε > 0。另一个主要成分是 Agarwal 和 Matousek [AM94] 的数据结构，用于回答射线射击查询，同样在 O(n1+ϵ/s1/4) 时间内。通过平衡时间和空间复杂度，Eppstein 和 Erickson 最终获得了一种直线骨架算法，该算法使用 O(n8/5+ϵ) 的时间和空间。他们进一步指出，该算法也适用于计算加权直线骨架。
For the unweighted case, Eppstein and Erickson [EE99] prove Theorem 2.7 in order to devise a slightly faster algorithm, see Section 2.1. In this case, all triangles of the algorithm have identical slope 1. This fact and the theorem mentioned above can be exploited in order to use more efficient data structures for intersection queries between rays and triangles. Using these refinements the algorithm by Eppstein and Erickson [EE99] has a theoretical worstcase time and space complexity of O(n1+ϵ + n8/11+ϵr9/11+ϵ) for the unweighted straight skeleton of planar straight-line graphs, where r denotes the number of reflex vertices.

对于未加权的情况，Eppstein和Erickson [EE99] 证明了定理2.7，以便设计一个稍微快一点的算法，见第2.1节。在这种情况下，算法的所有三角形都具有相同的斜率 1。这一事实和上述定理可以被利用，以便使用更有效的数据结构来进行射线和三角形之间的相交查询。使用这些改进，Eppstein和Erickson [EE99] 的算法在平面直线图的未加权直线骨架的情况下，具有 O(n1+ϵ + n8/11+ϵr9/11+ϵ) 的理论最坏情况时间和空间复杂度，其中 r 表示反射顶点的数量。
Eppstein and Erickson also present an approach that uses a simpler data structure by Eppstein [Epp00] in order to maintain the closest-pair between a ray and a triangle. This data structure uses a quadtree structure on the matrix of the pairwise distances. Using this data structure, the weighted straight skeleton can be computed in O(n log n + nr) time and O(nr) space and the unweighted straight skeleton can be computed in O(n log n + nr) time and O(n + r2) space. However, no implementations of these algorithms are known and, in particular, no runtime statistics have been published.

Eppstein和Erickson也提出了一种方法，该方法使用Eppstein [Epp00]提出的更简单的数据结构，以便维护射线和三角形之间的最近点对。这种数据结构在成对距离矩阵上使用四叉树结构。使用这种数据结构，加权直线骨架可以在O(n log n + nr)时间和O(nr)空间内计算，非加权直线骨架可以在O(n log n + nr)时间和O(n + r2)空间内计算。然而，这些算法的实现尚不为人所知，特别是，没有发布任何运行时统计数据。

Similar techniques as sketched above are also used by Eppstein and Erickson [EE99] in order to compute the motorcycle graph of n motorcycles m1, . . . , mn. The motorcycle graph was introduced in their paper in order to describe the essential sub problem of computing straight skeletons, see Section 1.2.4. The idea is that each motorcycle mi

defines a tilted ray in R3 and a vertical curtain above the ray. The intersection of a ray of the motorcycle mi with a curtain of a motorcycle mj corresponds to a crash of mi into the trace of mj. Their algorithm runs in O(n17/11+ϵ) time and space.

Eppstein和Erickson [EE99]也使用了如上所述的类似技术，以计算n辆摩托车m1, . . . , mn的摩托车图。摩托车图在他们的论文中被引入，以描述计算直线骨架的基本子问题，见第1.2.4节。其思想是，每辆摩托车mi在R3中定义了一条倾斜的光线以及光线上方的垂直幕帘。摩托车mi的光线与摩托车mj的幕帘的交点对应于mi撞入mj的轨迹。他们的算法在O(n17/11+ϵ)时间和空间内运行。
1.4.2.4 Cheng and Vigneron, 2002

1.4.2.4 Cheng和Vigneron，2002

Cheng and Vigneron [CV02, CV07] were the first to exploit the relationship between motorcycle graphs and straight skeletons in order to compute the straight skeleton. Their straightskeleton algorithm accepts non-degenerate polygons P with holes as input.

Cheng和Vigneron [CV02, CV07] 率先利用摩托车图和直线骨架之间的关系来计算直线骨架。他们的直线骨架算法接受带有孔的非退化多边形 P 作为输入。

First of all, Cheng and Vigneron compute the motorcycle graph M(P) induced by P. Let m1, . . . , mn denote n motorcycles. Further, we denote by pi the start point and by vi the velocity of the motorcycle mi. They observed that the tracks, on which the n motorcycles drive, can have Θ(n2) pairwise intersections, but only O(n) among them actually correspond a crash. The challenge is to geometrically separate motorcycles that do not interact. At the first sight this appears to be difficult, since a single motorcycle can be very fast and cross the tracks of many motorcycles. The central idea of Cheng and Vigneron is to employ so-called ϵ-cuttings.

首先，Cheng和Vigneron计算由P诱导的摩托车图M(P)。设m1, . . . , mn表示n辆摩托车。此外，我们用pi表示摩托车mi的起始点，用vi表示其速度。他们观察到，n辆摩托车行驶的轨迹可能具有Θ(n2)个成对交点，但其中只有O(n)个实际对应于碰撞。挑战在于以几何方式分离不相互作用的摩托车。乍一看，这似乎很困难，因为一辆摩托车可能非常快，并穿过许多摩托车的轨迹。Cheng和Vigneron的核心思想是采用所谓的ϵ-截断。

Chazelle [Cha04] explains ϵ-cuttings as follows. Let H denote a set of n hyperplanes in Rd. An ϵ-cutting is a tessellation of Rd into non-overlapping simplices such that each simplex intersects at most ϵn hyperplanes of H. Chazelle [Cha93] presented an algorithm that runs in optimal time O(nϵ1−d) and constructs a cutting of optimal size O(ϵ−d).

Chazelle [Cha04] 解释 ε-截断如下。设 H 表示 Rd 中的 n 个超平面的集合。一个 ε-截断是将 Rd 细分为不重叠的单纯形，使得每个单纯形最多与 H 的 εn 个超平面相交。Chazelle [Cha93] 提出了一个以最佳时间 O(nϵ1−d) 运行并构建最佳大小 O(ϵ−d) 的截断的算法。

Cheng and Vigneron [CV07] use 1/√n-cuttings in R2 on the supporting lines of the motorcycle tracks. Hence, every triangle of the cutting intersects at most √n motorcycle tracks. The cutting of O(n) size can be computed in O(n√n) time by Chazelle's algorithm

[Cha93]. Once the cutting is computed, Cheng and Vigneron [CV07] simulate the movement of the motorcycles within the cutting. They distinguish two types of events:

Cheng和Vigneron [CV07] 在摩托车轨迹的支撑线上使用1/√n-cuttings，位于R2中。因此，切割的每个三角形最多与√n条摩托车轨迹相交。大小为O(n)的切割可以通过Chazelle算法[Cha93]在O(n√n)时间内计算出来。一旦计算出切割，Cheng和Vigneron [CV07] 就会模拟摩托车在切割内的运动。他们区分了两种类型的事件：
• Switch event: A motorcycle reaches the boundary of a cutting-triangle and migrates to a neighboring triangle.

【NT0】•[/NT0] 切换事件：一辆摩托车到达切割三角形的边界并迁移到相邻的三角形。

• Crash20 event: A motorcycle crashes into the trace of another motorcycle.

【NT0】•【/NT0】Crash20 事件：一辆摩托车撞到另一辆摩托车的痕迹上。
The motorcycle graph can be computed independently within each cell. A priority queue Q contains each event and the events are processed in chronological order. First, they compute all potential switch events in advance and insert them all into the priority queue Q in O(n√n log n) time. Note that a single motorcycle may indeed cross ω(√n) simplices, but the total number of switch events among all motorcycles is in O(n√n).[21] In order to manage the potential crash events, Cheng and Vigneron maintain for each cutting-triangle an arrangement on the tracks of each motorcycle that is or has been visiting the triangle. Initially, each arrangement in a cutting triangle C is built for the motorcycles that start within C. Using a standard plane-sweep algorithm — e. g., Bentley-Ottmann [BO79] — this can be achieved in O(n√n log n) time.

摩托车图可以在每个单元格内独立计算。一个优先级队列 Q 包含每个事件，并且事件按时间顺序处理。首先，他们预先计算所有潜在的切换事件，并将它们全部插入到优先级队列 Q 中，时间复杂度为 O(n√n log n)。请注意，单个摩托车可能确实会穿过 ω(√n) 个单纯形，但所有摩托车之间的切换事件总数在 O(n√n).[21] 中。为了管理潜在的碰撞事件，Cheng 和 Vigneron 为每个切割三角形维护一个关于每辆正在或曾经访问该三角形的摩托车轨迹的排列。最初，切割三角形 C 中的每个排列都是为在 C 内启动的摩托车构建的。使用标准的平面扫描算法——例如，Bentley-Ottmann [BO79]——这可以在 O(n√n log n) 时间内实现。

When a switch event occurs for the motorcycle mi they first check whether mi has already crashed. If it is still driving they insert the track of mi into the arrangement of the new triangle C. Cheng and Vigneron use a binary tree structure on each cell in each arrangement, which enables them to insert the track of mi in O(k log n) time, where k denotes the number of new arrangement vertices introduced. For each such arrangement vertex they add a potential crash event into Q, depending on which of both motorcycles involved reaches the vertex earlier. Cheng and Vigneron argue that the total number of arrangement vertices is bounded by O(n√n). In the triangle C, they consider an arrangement vertex v on the intersection of the tracks of mi and mj. Either mi or mj started or crashed within C. Cheng and Vigneron charge mi with v, if mi started or crashed within C and likewise for mj. In order to bound the total number of arrangement vertices they

count for each motorcycle mk the number of vertices for which mk has been charged. In order to do so they only have to consider the cutting triangles where mk started resp. crashes. We get at most 2√n vertices for each motorcycle and O(n√n) in total. Hence, all switch events can be handled in O(n√n log n) time.

当摩托车mi发生切换事件时，他们首先检查mi是否已经发生碰撞。如果它仍在行驶，他们将mi的轨迹插入到新三角形C的排列中。Cheng和Vigneron在每个排列的每个单元格上使用二叉树结构，这使他们能够在O(k log n)时间内插入mi的轨迹，其中k表示引入的新排列顶点的数量。对于每个这样的排列顶点，他们将一个潜在的碰撞事件添加到Q中，这取决于所涉及的两辆摩托车中哪一辆先到达该顶点。Cheng和Vigneron认为，排列顶点的总数以O(n√n)为界。在三角形C中，他们考虑v处的一个排列顶点，该顶点位于mi和mj的轨迹的交点上。mi或mj在C内启动或发生碰撞。如果mi在C内启动或发生碰撞，则Cheng和Vigneron将mi与v相关联，mj的情况类似。为了限制排列顶点的总数，他们计算每辆摩托车mk被关联的顶点数。为了做到这一点，他们只需要考虑mk启动或发生碰撞的切割三角形。对于每辆摩托车，我们最多得到2√n个顶点，总共得到O(n√n)个顶点。因此，所有切换事件都可以在O(n√n log n)时间内处理。
When a crash event is processed for mi, Cheng and Vigneron check whether mi has not crashed already. If it is still driving, the motorcycle mi definitely crashed at the arrangement vertex for which the current crash event has been created. The total number of potential crash events is bounded by the total number of arrangement vertices, which is O(n√n), as elaborated above. A single crash event is easily handled in O(log n) time and hence the total costs for all crash events is in O(n√n log n).

当处理 mi 的碰撞事件时，Cheng 和 Vigneron 会检查 mi 是否尚未发生碰撞。如果它仍在行驶，则摩托车 mi 肯定在当前碰撞事件创建的排列顶点处发生了碰撞。潜在碰撞事件的总数受排列顶点总数的限制，如上所述，该值为 O(n√n)。单个碰撞事件可以很容易地在 O(log n) 时间内处理，因此所有碰撞事件的总成本为 O(n√n log n)。

The space complexity for all arrangements is bounded by O(n√n). Summarizing, Cheng and Vigneron compute the motorcycle graph of n motorcycles in O(n√n log n) time and O(n√n) space[22]. This is a slight improvement compared to the algorithm by Eppstein and Erickson [EE99].

所有排列的空间复杂度都以O(n√n)为界。总而言之，Cheng和Vigneron在O(n√n log n)的时间和O(n√n)的空间内计算n辆摩托车的摩托车图[22]。与Eppstein和Erickson [EE99]的算法相比，这是一个小小的改进。

Strictly speaking, the discussed algorithm computes M(m1, . . . , mn) but does not consider the edges of P as walls, i. e., they do not compute M(P) as defined in Section 1.2.4. However, Cheng and Vigneron note that their algorithm can handle motorcycles that run out of fuel when they reach the boundary of P. In order to bound the length of the motorcycle traces accordingly, one could apply the ray-shooting algorithm within a polygon by Chazelle et al. [CEG+91], which supports a single ray shooting in O(log n) time.

严格来说，所讨论的算法计算的是M(m1, . . . , mn)，但没有将P的边视为墙，也就是说，它们没有计算第1.2.4节中定义的M(P)。然而，Cheng和Vigneron指出，他们的算法可以处理摩托车在到达P边界时耗尽燃料的情况。为了相应地限制摩托车轨迹的长度，可以应用Chazelle等人提出的多边形内的光线投射算法[CEG+91]，该算法支持在O(log n)时间内进行单次光线投射。
Cheng and Vigneron [CV07] also mention a randomized and simpler version of their algorithm, where the supporting lines of the traces of √n randomly chosen motorcycles could be used instead of an 1/√n-cutting. Furthermore, they mention a simpler cutting algorithm in the plane by Har-Peled [HP00]. Nevertheless, no implementations or runtime statistics are known for their motorcycle graph algorithm.

Cheng和Vigneron [CV07] 还提到了他们算法的一个随机化且更简单的版本，其中可以使用随机选择的 √n 辆摩托车的轨迹的支撑线，而不是 1/√n-cutting。此外，他们还提到了Har-Peled [HP00] 提出的一个更简单的平面切割算法。然而，对于他们的摩托车图算法，目前尚无已知的实现或运行时统计数据。
In order to compute the straight skeleton of a simple non-degenerate polygon P with n vertices, Cheng and Vigneron [CV07] exploit Theorem 2.11. The fact that M(P) covers the reflex arcs of S (P) allows them to devise a randomized algorithm for the computation of S (P). In a nutshell, the algorithm chooses random sites on the straight skeleton which induce a partition of P. The idea is to compute S (P) on each part of the partition in a divideand-conquer fashion, as described in the following.

为了计算具有n个顶点的简单非退化多边形P的直线骨架，Cheng和Vigneron [CV07]利用了定理2.11。M(P)覆盖S (P)的反射弧这一事实使他们能够设计一种随机算法来计算S (P)。简而言之，该算法在直线骨架上选择随机点，这些点会引起P的划分。其思想是以分而治之的方式计算每个划分部分上的S (P)，如下所述。

They first choose a point p on a convex arc of S (P) and project p vertically onto the terrain T (P). The projection point pˆ lies on a ridge of T (P) and defines two or three paths on T (P) that follow the steepest descent and correspond to the raindrop traces in Section 1.2.3. Cheng and Vigneron consider a set E of such ridge points and consider the vertical projections of the corresponding raindrop paths onto the plane. These projected paths induce a partition of P, which Cheng and Vigneron call canonical partition of P induced by E. (Note that these paths do not cross, but may merge in a valley of T (P).) Cheng and Vigneron observed that the canonical partition of P induced by E can be computed recursively. Consider a polygon that is partition by E1 and a cell C of that partition that is further partitioned by E2. They proved that the resulting partition is equal to the partition of P induced by E1 ∪ E2.

他们首先在S (P)的凸弧上选择一个点p，然后将p垂直投影到地形T (P)上。投影点pˆ位于T (P)的脊线上，并在T (P)上定义了两条或三条路径，这些路径遵循最陡下降方向，对应于第1.2.3节中的雨滴轨迹。Cheng和Vigneron考虑了这样一组脊点E，并考虑了相应雨滴路径在平面上的垂直投影。这些投影路径导致了P的一个划分，Cheng和Vigneron称之为由E引起的P的规范划分。（请注意，这些路径不会交叉，但可能会在T (P)的山谷中合并。）

Cheng和Vigneron观察到，由E引起的P的规范划分可以递归计算。考虑一个被E1划分的多边形，以及该划分的一个单元C，该单元又被E2进一步划分。他们证明了最终的划分等于由E1 ∪ E2引起的P的划分。
Cheng and Vigneron presented a method to compute the intersection points L ∩ S (P), for a line L parallel to the y-axis, by computing a vertical slice of T (P) above L. The resulting points E are used in order to subdivide P according to the partition scheme described above. Cheng and Vigneron keep on subdividing cells of the partition scheme using this method. Once the cells have reached a certain size, they compute the straight skeleton on this cell by a brute force method. The way how Cheng and Vigneron select the vertical line L involves randomness. Finally, Cheng and Vigneron are able to compute the straight skeleton of a simple non-degenerate polygon P in expected O(n log2 n) time if the motorcycle graph is already given. Since the motorcycle graph M(P) can be computed in O(r√r log r) time they obtain a straight-skeleton algorithm that runs in O(n log2 n + r√r log r) time.

Cheng和Vigneron提出了一种计算直线L ∩ S (P)交点的方法，其中直线L平行于y轴，通过计算L上方T (P)的垂直切片来实现。得到的点E用于根据上述分割方案细分P。Cheng和Vigneron继续使用这种方法细分分割方案的单元格。一旦单元格达到一定大小，他们就通过蛮力法计算该单元格上的直线骨架。Cheng和Vigneron选择垂直线L的方式涉及随机性。最后，如果已经给出了摩托车图，Cheng和Vigneron能够在期望的O(n log2 n)时间内计算出一个简单非退化多边形P的直线骨架。由于摩托车图M(P)可以在O(r√r log r)时间内计算出来，他们获得了一个在O(n log2 n + r√r log r)时间内运行的直线骨架算法。

Cheng and Vigneron [CV07] extended their algorithm to non-degenerate polygons with holes. If h denotes the number of holes their algorithm computes the straight skeleton in O(n√h log2 n + r√r log r) expected time.

Cheng和Vigneron [CV07] 将他们的算法扩展到具有孔的非退化多边形。如果 h 表示孔的数量，他们的算法在 O(n√h log2 n + r√r log r) 预期时间内计算出直线骨架。

1.4.2.5 Felkel and Obdržálek, 1999

1.4.2.5 Felkel和Obdržálek，1999

Felkel and Obdržálek [FO99] briefly presented a wavefront-type straight-skeleton algorithm for simple polygons that is based on the algorithm of Aichholzer et al. [AAAG95]. As discussed in Section 1.4.2.1, the simulation of the edge events is simple, but handling split events is the challenge for a wavefront-type straight-skeleton algorithm.

Felkel和Obdržálek [FO99] 简要介绍了一种用于简单多边形的波前型直线骨架算法，该算法基于Aichholzer等人 [AAAG95] 的算法。正如第1.4.2.1节所讨论的，边事件的模拟很简单，但是处理分裂事件是波前型直线骨架算法的挑战。

They again maintain a priority queue of edge events and split events. Felkel and Obdržálek [FO99] propose the following procedure in order to determine the split event for a reflex vertex v of P. For each edge e they determine a point pe on the bisector ray emanated from v which has equal orthogonal distance to e and the incident edges of v.

The edge e defines a triangle ∆e (possibly infinite) that is bounded by e and the bisector rays emanated from the incident vertices of e. If pe does not lie in ∆e then they ignore pe. Among all edges e of P, with pe lying in ∆e, they select the one point pe whose distance from v is smallest. They add a split event for v at this point into our priority queue.

他们再次维护一个边事件和分割事件的优先级队列。Felkel 和 Obdržálek [FO99] 提出了以下程序，以确定 P 的一个反射顶点 v 的分割事件。对于每条边 e，他们确定在从 v 发出的平分射线上的一个点 pe，该点到 e 的正交距离与到 v 的相邻边的正交距离相等。边 e 定义了一个三角形 ∆e（可能是无限的），该三角形由 e 和从 e 的相邻顶点发出的平分射线所界定。如果 pe 不在 ∆e 中，那么他们忽略 pe。在 P 的所有边 e 中，如果 pe 位于 ∆e 中，他们选择距离 v 最近的点 pe。他们将 v 在此点的分割事件添加到我们的优先级队列中。
This algorithm takes O(nr + n log n) time in order to compute the straight skeleton. Unfortunately, the procedure which computes the next split event does not necessarily determine the correct split event. In Figure 13 we give an example where the split event of a vertex v is not computed correctly. Different issues concerning the correctness are mentioned in Yakersberg [Yak04].

该算法需要 O(nr + n log n) 的时间来计算直线骨架。不幸的是，计算下一个分割事件的过程不一定能确定正确的分割事件。在图 13 中，我们给出了一个顶点 v 的分割事件未被正确计算的例子。Yakersberg [Yak04] 提到了关于正确性的不同问题。

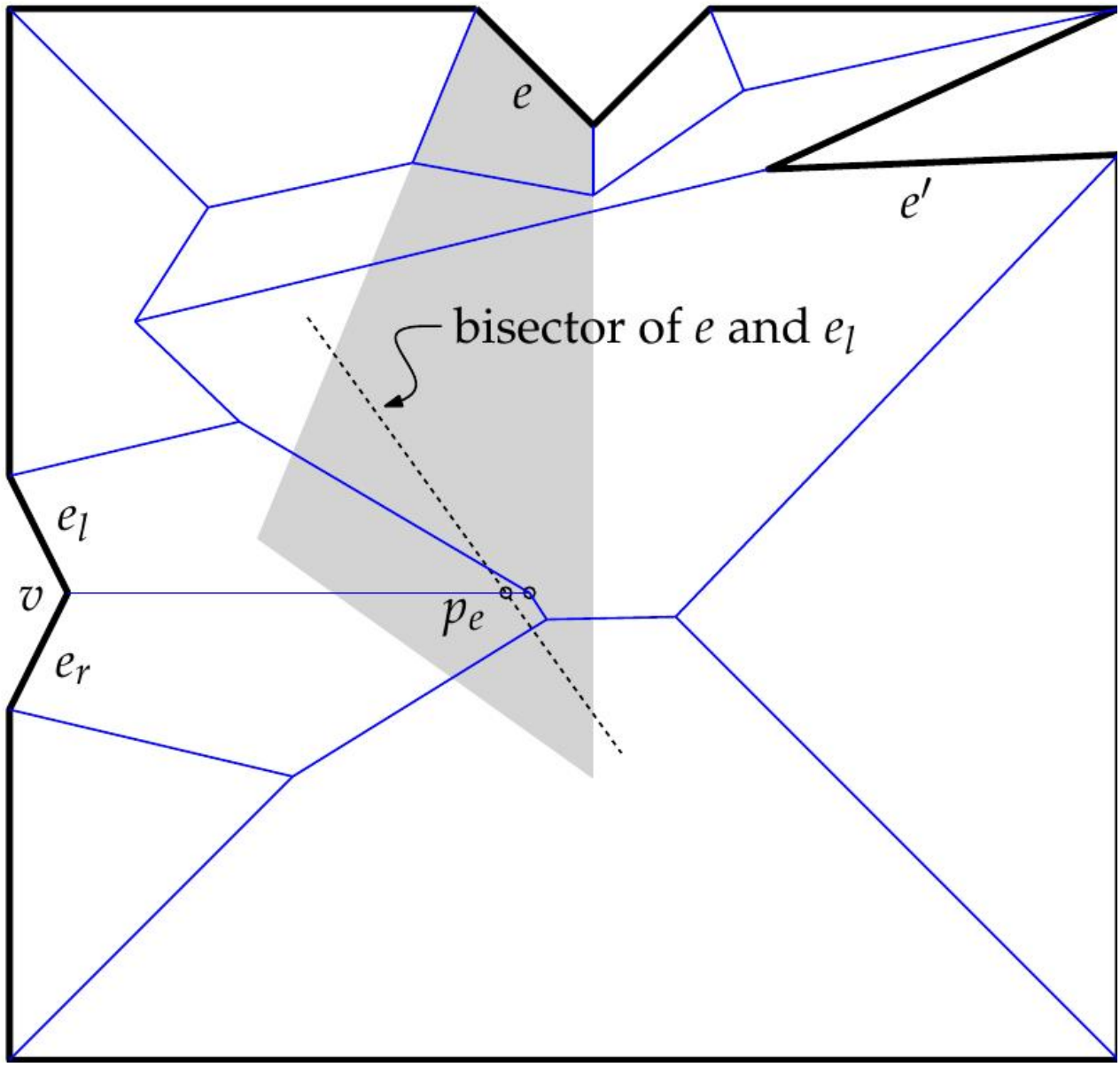


Figure 13: The point pe does not mark the split event of the reflex vertex v. The vertex v actually causes a crash with the edge e′, which happens after v passed pe.

图13：点pe并未标记反射顶点v的分裂事件。顶点v实际上导致与边e′发生碰撞，这发生在v经过pe之后。

1.4.3 Implementations

1.4.3 实现

To the best of our knowledge the only published runtime statistics are due to Felkel and Obdržálek [FO99], who published the runtime consumption of their code for seven datasets. Beside the implementation of Felkel and Obdržálek, the only implementation available is the straight-skeleton code by Cacciola [Cac04] that is shipped with CGAL [CGA]. The algorithm behind the CGAL implementation was originally based on the algorithm by Felkel and Obdržálek [FO99]. However, Cacciola has significantly adapted the original algorithm in order to get it to work correctly.[23] The current version, CGAL 3.8, can compute the straight skeleton of polygons with holes. Unfortunately, no details on the

implemented algorithm are published. In Section 2.5.4, we present experimental results that yield an O(n2 log n) runtime performance and an O(n2) memory footprint for the CGAL implementation.

据我们所知，唯一已发表的运行时统计数据来自Felkel和Obdržálek [FO99]，他们发表了他们的代码在七个数据集上的运行时消耗。除了Felkel和Obdržálek的实现之外，唯一可用的实现是由Cacciola [Cac04]提供的直线骨架代码，该代码随CGAL [CGA]一起发布。CGAL实现背后的算法最初基于Felkel和Obdržálek [FO99]的算法。然而，Cacciola对原始算法进行了重大调整，以使其能够正确工作。[23] 当前版本CGAL 3.8可以计算带孔多边形的直线骨架。遗憾的是，没有公布关于已实现算法的详细信息。在第2.5.4节中，我们展示了实验结果，这些结果表明CGAL实现的运行时性能为O(n2 log n)，内存占用为O(n2)。

1.4.4 Summary

1.4.4 摘要

The best known lower bounds for the straight skeleton of simple polygons is Ω(n) resp. Ω(n log n) for simple polygons with holes and planar straight-line graphs. This poses a significant gap to the fastest algorithms known so far, see Table 1. The fastest algorithm is due to Eppstein and Erickson [EE99] with a runtime of O(n1+ϵ + n8/11+ϵr9/11+ϵ) for planar straight-line graphs. For simple non-degenerate polygons Cheng and Vigneron [CV07] presented an algorithm with a slightly better expected runtime of O(n log2 n + r√r log r).

对于简单多边形的直线骨架，最著名的下界是Ω(n)。对于带孔的简单多边形和平面直线图，下界是Ω(n log n)。这与目前已知的最快算法之间存在显著差距，参见表1。最快的算法归功于Eppstein和Erickson [EE99]，对于平面直线图，其运行时间为O(n1+ϵ + n8/11+ϵr9/11+ϵ)。对于简单的非退化多边形，Cheng和Vigneron [CV07] 提出了一种算法，其预期运行时间略好，为O(n log2 n + r√r log r)。

Algorithm	Time	Space	PSLG	Impl.
[AAAG95]	$O(nr \log n)$	$O(n)$	No	Yes
[AA96]	$O(n^3 \log n)$	$O(n)$	Yes	Yes
[EE99]	$O(n^{1+\epsilon} + n^{8/11+\epsilon} r^{9/11+\epsilon})$	$O(n^{1+\epsilon} + n^{8/11+\epsilon} r^{9/11+\epsilon})$	Yes	No
[CV02]	exp. $O(n \log^2 n + r\sqrt{r} \log r)$		No	No

Table 1: Summary of known straight-skeleton algorithms and their time and space complexities. The input contains n vertices and r denotes the number of reflex wavefront vertices. The column “PSLG” denotes whether the algorithm accepts a planar straight-line graph as input. The column “Impl.” denotes whether the algorithm is suitable for implementation from a practical point of view.

表 1：已知的直骨架算法及其时间和空间复杂度总结。输入包含 n 个顶点，r 表示反射波前顶点的数量。“PSLG”列表示该算法是否接受平面直线图作为输入。“Impl.”列表示该算法是否适合从实际角度进行实现。

Algorithm	Time	Space	Impl.
[EE99]	$O(n^{17/11+\epsilon})$	$O(n^{17/11+\epsilon})$	No
[CV02]	$O(n\sqrt{n}\log n)$	$O(n\sqrt{n})$	No

Table 2: Summary of known motorcycle-graph algorithms and their time and space complexities for n motorcycles. The column “Impl.” denotes whether the algorithm is suitable for implementation from a practical point of view.

表2：已知摩托车图算法及其对于n辆摩托车的时间和空间复杂度总结。“Impl.”列表示该算法是否适合从实践角度进行实施。

On the implementation side there is the straight-skeleton code by CGAL which accepts polygons with holes as input. It is a wavefront-type algorithm which is roughly based on the algorithms due to [AAAG95, FO99]. The only straight-skeleton algorithm which is suitable for implementation and accepts planar straight-line graphs as input was presented by Aichholzer and Aurenhammer [AA98]. However, no implementation is available and no runtime statistics have been published.

在实现方面，CGAL 提供了直接骨架代码，该代码接受带孔多边形作为输入。它是一种波前型算法，大致基于 [AAAG95, FO99] 中的算法。唯一适用于实现并接受平面直线图作为输入的直接骨架算法由 Aichholzer 和 Aurenhammer [AA98] 提出。然而，没有可用的实现，也没有发布运行时统计数据。

1.5 g e n e r a l i z a t i o n s a n d r e l a t e d p r o b l e m s

1.5 推广与相关问题

1.5.1 Linear axis

1.5.1 线性轴

The linear axis is a variation of the straight skeleton for simple polygons P introduced by Tă- nase and Veltkamp [TV04a]. The difference between the linear axis and the straight skeleton is a more general definition of the initial wavefront at reflex vertices of P.

线性轴是简单多边形P的直线骨架的一种变体，由Tă-nase和Veltkamp [TV04a] 提出。线性轴和直线骨架之间的区别在于P的凹顶点处的初始波前的更一般的定义。
The idea is that a reflex vertex v of P does not emanate a single reflex wavefront vertex but k reflex wavefront vertices v1, . . . , vk, with 1 ≤ k. The wavefront edges connecting two consecutive vertices vi, vi+1 propagate with unit speed and for each vi the two incident edges span identical angles, see Figure 14. The skeleton structure which results from this initial wavefront and the ordinary straight-skeleton wavefront propagation is

called linear axis. For k = 1 the linear axis is identical to the straight skeleton. Denoting by α ∈ [π, 2π) the interior angle at the vertex v of P, the speeds of the reflex wavefront vertices are given by

其思想是，P 的一个凹顶点 v 不会发出单个凹波前顶点，而是发出 k 个凹波前顶点 v1, . . . , vk，其中 1 ≤ k。连接两个连续顶点 vi, vi+1 的波前边缘以单位速度传播，并且对于每个 vi，两个入射边缘跨越相同的角度，参见图 14。由此初始波前和普通直骨架波前传播产生的骨架结构称为线性轴。Fork = 1 时，线性轴与直骨架相同。用 α ∈ [π, 2π) 表示顶点 v 在 P 处的内角，凹波前顶点的速度由下式给出

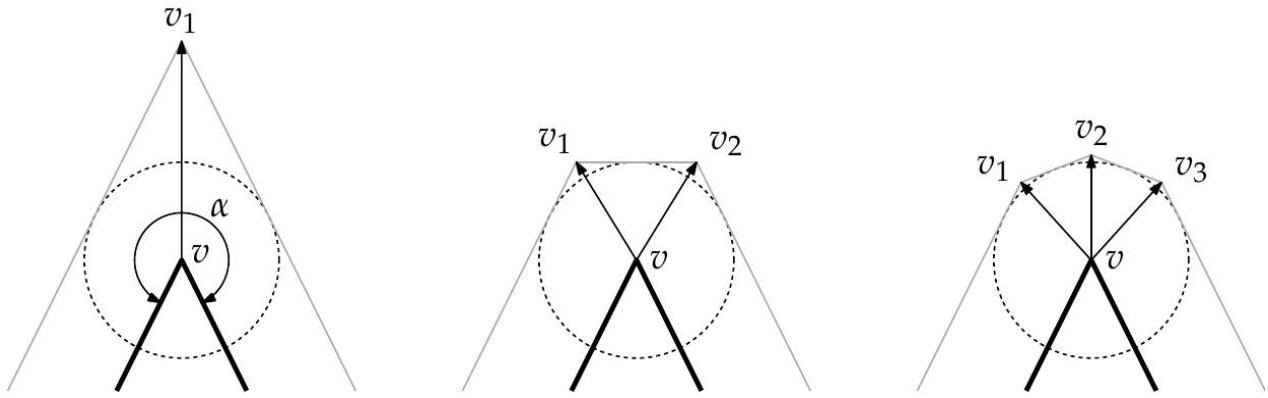


Figure 14: The wavefront (gray) of the linear axis at a reflex vertex v of the input polygon (bold) for different numbers k of emanated wavefront vertices v1, . . . , vk, with k = 1, 2, 3. The angles between consecutive wavefront edges are equal.

图 14：对于输入多边形（粗体）的反射顶点 v 处，线性轴的波前（灰色），对应于不同数量 k 的发出的波前顶点 v1, . . . , vk，其中 k = 1, 2, 3。连续波前边缘之间的角度相等。

$$\frac{1}{\cos\frac{\alpha-\pi}{2k}}. \tag{1.3}$$

Note that this expression tends towards 1 for large k. In particular, the speeds are bounded by √2 for k ≥ 2, whereas for k = 1 the reflex wavefront vertices can get arbitrarily fast when α gets close to 2π. The larger k gets, the better the wavefront approximates a circular arc. Tănase and Veltkamp [TV04a] proved that the linear axis converges to the medial axis as k grows for all reflex vertices. The concept of the linear axis has later been generalized to planar straight-line graphs by Vyatkina [Vya09a].

注意，对于较大的k，此表达式趋向于1。特别地，对于k ≥ 2，速度以√2为界，而对于k = 1，当α接近2π时，反射波前顶点可以任意快。k越大，波前越接近于圆弧。Tănase和Veltkamp [TV04a]证明，对于所有反射顶点，当k增长时，线性轴收敛到中轴。Vyatkina [Vya09a]后来将线性轴的概念推广到平面直线图。

Since the linear axis follows the same wavefront propagation process as the straight skeleton, one can basically apply the algorithms known for straight skeletons in order to compute the linear axis. For k ∈ O(1) at all reflex vertices, we obtain the same time and space complexities. However, Tănase and Veltkamp showed that if k is chosen large enough such that the medial axis and the linear axis differ in a sufficiently small amount then the linear axis can be obtained from the medial axis in linear time.

由于线性轴遵循与直线骨架相同的波前传播过程，因此基本上可以应用已知的直线骨架算法来计算线性轴。对于所有凹顶点处的 k ∈ O(1)，我们获得相同的时间和空间复杂度。然而，Tănase 和 Veltkamp 表明，如果选择 k 足够大，使得中轴线和线性轴的差异足够小，那么可以在线性时间内从中轴线获得线性轴。

Of course, we can further generalize the linear axis by not requiring that consecutive wavefront edges in Figure 14 have identical angles. Basically, we can emanate any polygonal chain as wavefront as long as the wavefront edges have orthogonal distance to v that is equal to the time elapsed so far. We also refer to the discussion in Section 1.2.2 on various types of wavefront polygons emanated at isolated vertices and terminal vertices of the input graph.

当然，我们可以进一步推广线性轴，而不要求图14中连续波前边缘具有相同的角度。基本上，我们可以将任何多边形链作为波前发出，只要波前边缘到v的正交距离等于目前为止所经过的时间。我们还可以参考第1.2.2节中关于在输入图的孤立顶点和终端顶点发出的各种类型的波前多边形的讨论。

The advantages of the linear axis compared to the straight skeleton are all, more or less, related to a reduced speed of the reflex vertices. From a practical point of view, we expect that the linear axis is easier to compute in terms of numerical stability. Tănase and Veltkamp [TV04a] mention that the polygon decomposition application from Section 1.3.5 would also benefit if the speed of the reflex wavefront vertices is limited. Finally, we want to review the mitered offset application for NC-machining, see Section 1.3.1. A tool path based on the straight skeleton can lead to long paths at very sharp reflex vertices. The offset curve based on the linear axis would give us a shorter path while still avoiding a continuous contact of the tool with reflex vertices, as it would be the case for the medial axis.

与直线骨架相比，线性轴的优势或多或少都与反射顶点的速度降低有关。从实践的角度来看，我们期望线性轴在数值稳定性方面更容易计算。Tănase和Veltkamp [TV04a] 提到，如果反射波前顶点的速度受到限制，第1.3.5节中的多边形分解应用也将受益。最后，我们想回顾一下用于数控加工的斜接偏移应用，见第1.3.1节。基于直线骨架的刀具路径可能导致非常尖锐的反射顶点处出现较长的路径。基于线性轴的偏移曲线将为我们提供更短的路径，同时仍然避免刀具与反射顶点的连续接触，就像中轴线的情况一样。
1.5.2 Weighted straight skeleton

1.5.2 加权直线骨架

Eppstein and Erickson [EE99] were the first to mention the weighted straight skeleton, where the wavefront edges are allowed to propagate at different speeds. The geometry of the weighted straight skeleton has not yet been systematically investigated and, in particular, its relation to the motorcycle graph is unclear. However, one can observe that basic properties of the unweighted straight skeleton do not carry over to the weighted case. For example, in the unweighted case every straight-skeleton face is monotone w. r. t. its defining wavefront edge, cf. Lemma 2.3 in Section 2.1. As Figure 15 (a) illustrates, this is not necessarily the case for the weighted straight skeleton. Moreover, important theorems concerning alternative characterizations of the straight skeleton do not hold for the weighted case. Eppstein and Erickson [EE99] presented a simple polygon for which Theorem 2.7 does not hold in the weighted case, see Figure 16. The same figure also serves as a counter-example to Theorem 2.11.

Eppstein和Erickson [EE99] 首先提到了加权直线骨架，其中波前边缘可以以不同的速度传播。加权直线骨架的几何性质尚未得到系统研究，特别是它与摩托车图的关系尚不清楚。然而，可以观察到，未加权直线骨架的基本性质不能直接推广到加权情况。例如，在未加权情况下，每个直线骨架面相对于其定义的波前边缘是单调的，参见第2.1节中的引理2.3。如图15(a)所示，对于加权直线骨架，情况不一定是这样。此外，关于直线骨架的替代特征的重要定理不适用于加权情况。Eppstein和Erickson [EE99] 提出了一个简单多边形，对于该多边形，定理2.7在加权情况下不成立，参见图16。同一张图也可以作为定理2.11的反例。

Beside the fact that certain properties do not carry over from the unweighted straight skeleton, we observe that the definition of the weighted straight skeleton is tricky in the presence of parallel input segments. Figure 15 shows two wavefront edges e1, e2 that propagate with unit speed and e3 that propagates with speed 2. Let us rotate the edge e1 around its right endpoint such that e1 and e3 become parallel in both subfigures. In the limit we obtain two identical inputs in both subfigures, but in subfigure (a) the straight-skeleton arc between e1 and e3 is pointing to the left, whereas in subfigure (b) this arc is pointing to the right. This example poses a singularity in the naive wavefront-based definition of the weighted straight skeleton. Also note that in subfigure (a) the arc between e1 and e3 is convex, but the arc encloses an angle greater than $\pi/2$ with e3. Analogously for subfigure (b), where this arc is reflex but e3 and the arc encloses an angle less than $\pi/2$. For the unweighted straight skeleton, however, we could characterize reflex and convex arcs by the angle enclosed with its defining wavefront edges.

除了某些性质不能从无权直线骨架继承之外，我们观察到在存在平行输入线段的情况下，加权直线骨架的定义很棘手。图 15 显示了两个以单位速度传播的波前边缘 e1, e2 和一个以速度 2 传播的 e3。让我们绕其右端点旋转边缘 e1，使得在两个子图中 e1 和 e3 变得平行。在极限情况下，我们在两个子图中获得两个相同的输入，但在子图 (a) 中，e1 和 e3 之间的直线骨架弧指向左侧，而在子图 (b) 中，该弧指向右侧。这个例子在加权直线骨架的朴素波前定义中提出了一个奇点。另请注意，在子图 (a) 中，e1 和 e3 之间的弧是凸的，但该弧与 e3 围成的角度大于 π/2。与子图 (b) 类似，其中该弧是凹的，但 e3 和该弧围成的角度小于 π/2。然而，对于无权直线骨架，我们可以通过与其定义波前边缘围成的角度来表征凹弧和凸弧。

Lemma 1.11. Let v denote a wavefront vertex incident to two edges ea and eb, where ea propagates with speed a > 0 and eb propagates with speed b > 0. Further assume that ea and eb span an angle of α ∈ (0, 2π) at the side where the wavefront propagates to. Then the trajectory of v encloses with eb an angle αb and the wavefront vertex v has a speed of sin1αb , where

引理 1.11. 设 v 表示一个入射到两条边 ea 和 eb 的波前顶点，其中 ea 以速度 a > 0 传播，eb 以速度 b > 0 传播。进一步假设 ea 和 eb 在波前传播的一侧跨越一个角度 α ∈ (0, 2π)。则 v 的轨迹与 eb 围成一个角度 αb，并且波前顶点 v 的速度为 sin1αb，其中

$$\alpha_b = \frac{\pi}{2} - \arctan\cos\frac{\alpha + a_b}{b}$$

sin

α

.

(1.4)

Proof. Let us consider Figure 17. Simple trigonometry leads to x = −a · sin α, y = a · cos α, z = tan(b2+πy−α) and finally to αb = π/2 + arctan x+b z .

证明。 让我们考虑图17。简单的三角学推导出 x = −a · sin α, y = a · cos α, z = tan(b2+πy−α)，最后得到αb = π/2 + arctan x+b z。

In the unweighted case, i. e. a = b, we obtain αb = α/2 as expected. If we consider a/b fixed but less than 1 we observe the following behavior of αb at α = π.

在未加权的情况下，即 a = b，我们得到 αb = α/2，正如预期的那样。如果我们考虑 a/b 是固定的但小于 1，我们观察到 αb 在 α = π 时的以下行为。

lim

α

↗

π

α

b

=

π

lim

α

↘

π

α

b

=

0.

(1.5)

Both equations correspond to the two cases illustrated in Figure 15, with v denoting the wavefront vertex on the arc between e1 and e3 and eb = e3. The left equation is related to Figure 15 (a) while the right equation describes the situation in Figure 15 (b).

这两个方程对应于图15中所示的两种情况，其中v表示e1和e3之间弧上的波前顶点，且eb = e3。左边的方程与图15（a）相关，而右边的方程描述了图15（b）中的情况。

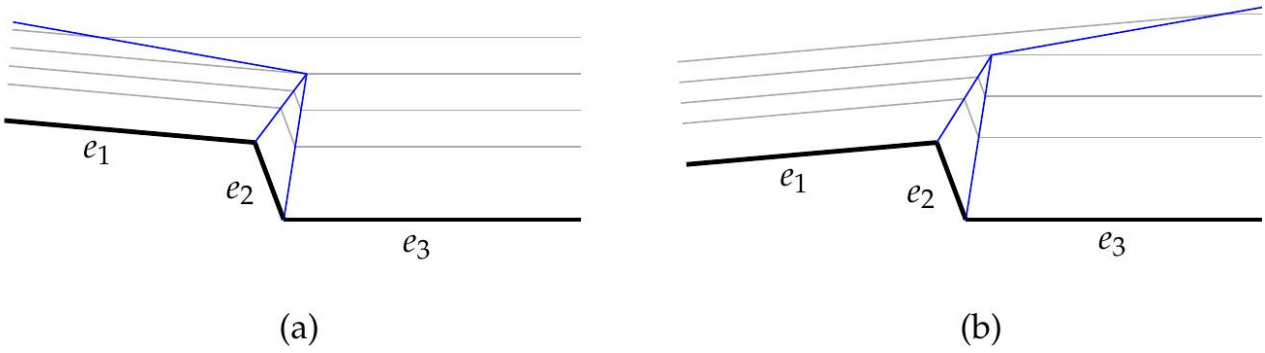


Figure 15: The weighted straight skeleton of three input edges. The edges e1 and e2 propagate at unit speed, but e3 propagates with speed 2. If we rotate e1 such that e1 and e3 become parallel, we obtain two different straight skeletons in the limit for the two subfigures.

图15：三个输入边的加权直线骨架。e1和e2以单位速度传播，但e3以速度2传播。如果我们旋转e1使得e1和e3变得平行，那么在极限情况下，对于这两个子图，我们会得到两个不同的直线骨架。

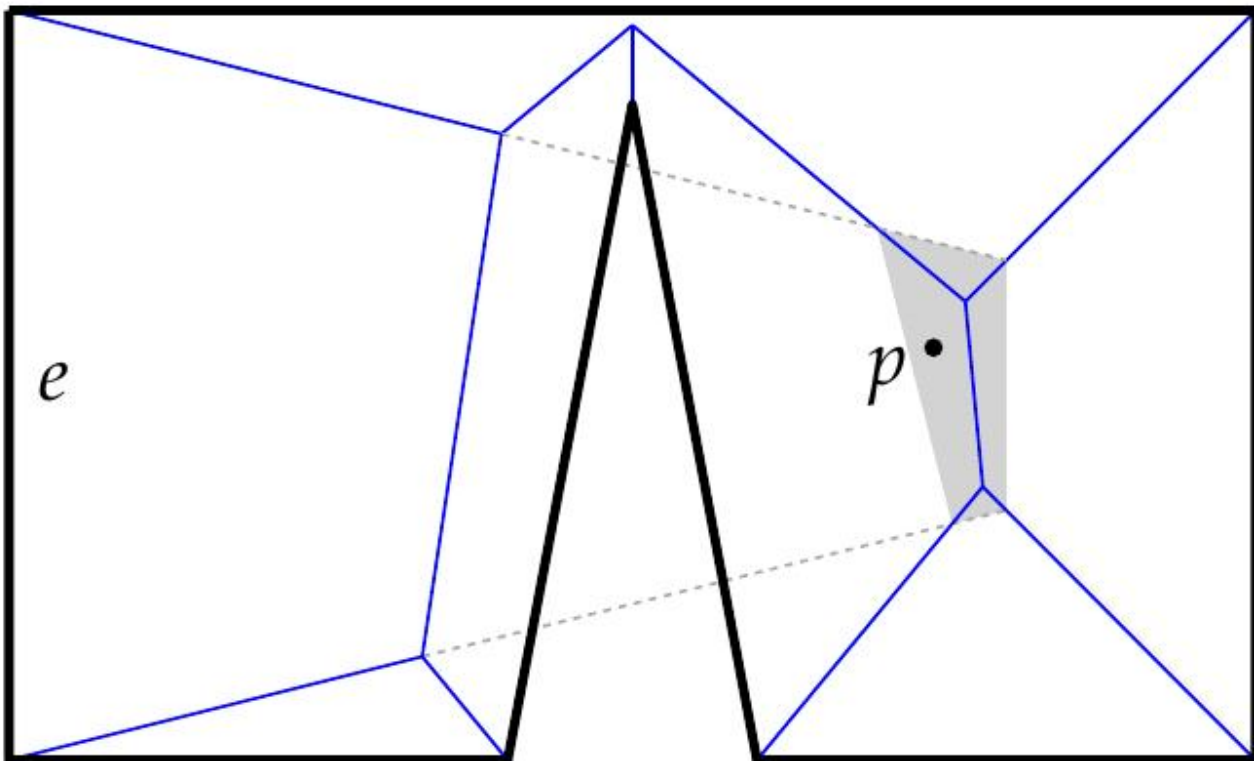


Figure 16: Assume all edges propagate with unit speed, but e has speed 4. Then the edge slab of e cuts off the shaded area from the terrain, i. e. the edge slab reaches below T (P) at point p. (This figure is based on Figure 6 in [EE99].)

图 16：假设所有边都以单位速度传播，但 e 的速度为 4。那么 e 的边板会从地形中切掉阴影区域，即边板在点 p 处到达 T (P) 的下方。（此图基于 [EE99] 中的图 6。）

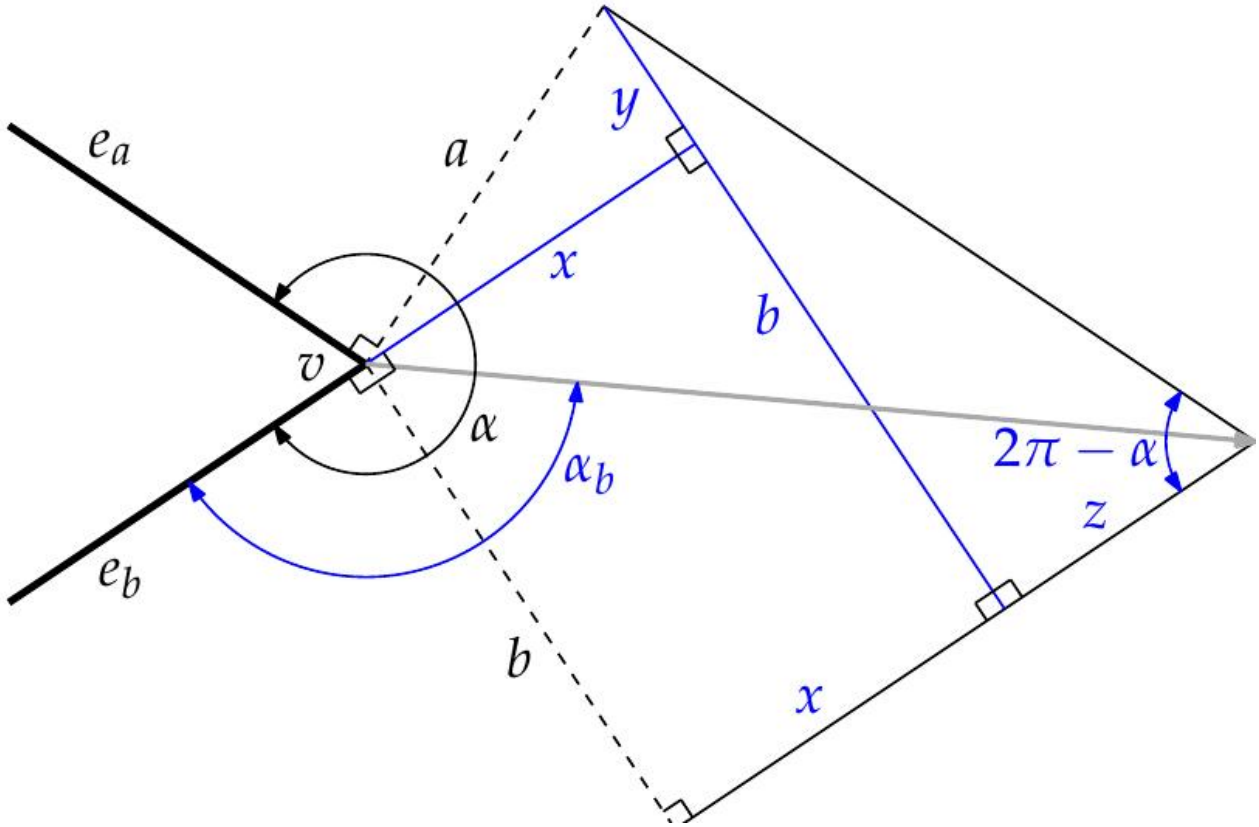


Figure 17: The speed and direction (gray arrow) of a vertex v which is incident to two edges ea and eb. The edge ea is propagating with speed a and eb is propagating with speed b.

图 17：顶点 v 的速度和方向（灰色箭头），该顶点与两条边 ea 和 eb 相关联。边 ea 以速度 a 传播，而 eb 以速度 b 传播。

As mentioned in Section 1.4.2.3, Eppstein and Erickson [EE99] presented an algorithm for the weighted straight skeleton of planar straight-line graphs with n vertices which uses O(n8/5+ϵ) time and space.

正如第1.4.2.3节所述，Eppstein和Erickson [EE99] 提出了一个算法，用于计算具有n个顶点的平面直线图的加权直线骨架，该算法使用O(n8/5+ϵ)的时间和空间。

1.5.3 Straight skeleton of polyhedra in R3

1.5.3 R3中多面体的直骨架

Straight skeletons of three-dimensional polyhedra have first been mentioned by Demaine et al. [DDLS05], who investigated so-called hinged dissections of polyhedra. A dissection of two polyhedra is a tessellation of the first polyhedron into polyhedral pieces such that

the second polyhedron can be built by the pieces of the first one. Demaine et al. [DDLS05] discuss so-called hinged dissections which are special dissections where the polyhedral pieces are hinged together at points or edges. It is an open question whether there exists a hinged dissection for every pair of polyhedra. However, the straight skeleton of polyhedra is used by Demaine et al. in order to compute hinged dissections for a certain class of polyhedra.

三维多面体的直骨架最早由 Demaine 等人 [DDLS05] 提及，他们研究了所谓的多面体铰链解剖。两个多面体的解剖是指将第一个多面体细分为多面体块，使得第二个多面体可以通过第一个多面体的块来构建。Demaine 等人 [DDLS05] 讨论了所谓的铰链解剖，这是一种特殊的解剖，其中多面体块在点或边上铰接在一起。对于每一对多面体是否存在铰链解剖，这是一个悬而未决的问题。然而，Demaine 等人使用多面体的直骨架来计算某一类多面体的铰链解剖。

The straight skeleton S (P) of a three-dimensional polyhedron P is defined by a wavefront propagation process, where the faces of P move inwards with unit speed in a self-parallel fashion. The set of points which are traced by the intersections of two adjacent faces is called the straight skeleton. Demaine et al. mention that S (P) gives a decomposition of P into cells such that one cell belongs to exactly one face of P. Further, Demaine et al. cite a personal communication with J. Erickson, in which it is mentioned that the straight skeleton in R3 is no longer uniquely defined, because at certain points in time one can make multiple decisions to continue the offset propagation.

三维多面体P的直骨架S(P)由波前传播过程定义，其中P的面以单位速度向内自平行移动。由两个相邻面的交点追踪的点集称为直骨架。Demaine等人提到，S(P)将P分解为若干个单元，使得一个单元恰好属于P的一个面。此外，Demaine等人引用了与J. Erickson的私人交流，其中提到R3中的直骨架不再是唯一确定的，因为在某些时间点，人们可以做出多个决定来继续偏移传播。

Barequet et al. [BEGV08] were the first to investigate algorithms for the straight skeleton of polyhedra P of certain types. Firstly, they mention a lower bound of Ω(n2) for the size of S (P) of polyhedra with n vertices. Note that the medial axis and the straight skeleton are identical for a convex polyhedron and the lower bound of Ω(n2) due to Held [Hel94] can be applied. Secondly, Barequet et al. mention O(n3+ϵ) as the best known upper bound due to Sharir [Sha94].

Barequet等人[BEGV08]率先研究了某些类型多面体P的直线骨架算法。首先，他们提到，对于具有n个顶点的多面体，S (P)的大小存在Ω(n2)的下界。请注意，对于凸多面体，中轴和直线骨架是相同的，因此可以应用Held [Hel94]提出的Ω(n2)下界。其次，Barequet等人提到Sharir [Sha94]提出的O(n3+ϵ)是已知的最佳上界。

Furthermore, Barequet et al. [BEGV08] presented a straight-skeleton algorithm for a certain class of polyhedra. First, they start with polyhedra made of voxels, i. e. axis-parallel cubes of identical size. Second, an extension to polyhedra with axis-parallel edges is presented. In the latter case the complexity of the straight skeleton can be bound to O(n2). Barequet et al. [BEGV08] present two algorithms for n-vertex polyhedra

with axis-parallel edges. The first runs in O(n2 log n) time, the second in O(k logO(1) n) time, where k denotes the size of the straight skeleton. Putting both algorithms together results in a runtime of O(min{n2 log n, k logO(1) n}).

此外，Barequet等人[BEGV08]提出了一种针对特定类型多面体的直骨架算法。首先，他们从由体素组成的多面体开始，即大小相同的轴平行立方体。其次，提出了一个扩展到具有轴平行边的多面体的方法。在后一种情况下，直骨架的复杂度可以限制在O(n2)。Barequet等人[BEGV08]提出了两种针对具有轴平行边的n顶点多面体的算法。第一种算法在O(n2 log n)时间内运行，第二种算法在O(k logO(1) n)时间内运行，其中k表示直骨架的大小。将两种算法结合起来，得到的运行时间为O(min{n2 log n, k logO(1) n})。
An interesting result of Barequet et al. [BEGV08] is that general simple n-vertex polyhedra exist, where the complexity of the straight skeleton is super-quadratic, namely Ω(n2α2(n)), where α(.) denotes the extremely slowly growing inverse Ackermann function. Furthermore, Barequet et al. shed more light on the ambiguity issue of straight skeletons of polyhedra.

Barequet等人[BEGV08]的一个有趣结果是，存在一般的简单n顶点多面体，其中直线骨架的复杂度是超二次的，即Ω(n2α2(n))，其中α(.)表示增长极其缓慢的反阿克曼函数。此外，Barequet等人进一步阐明了多面体直线骨架的歧义性问题。

1.5 generalizations and related problems 33

1.5.4 City Voronoi diagrams

1.5.4 城市Voronoi图

The straight skeleton was used by Aichholzer et al. [AAP04] in order to compute the socalled city Voronoi diagram which is presented as follows. Let C denote a planar straightline graph with c vertices and edges that are axis-parallel. The graph C models a public transit system like the subway. The problem is to find the shortest path between two vertices x andy in the plane, where the distance measure on R2 \ C is given by the Manhattan metric. Furthermore, one may enter and leave the public transit at any point of C with zero delay. The public transit is assumed to move on the edges of C with speed v > 1.

Aichholzer等人[AAP04]使用了直线骨架来计算所谓的城市Voronoi图，其表述如下。设C表示一个具有c个顶点和边的平面直线图，这些顶点和边与坐标轴平行。图C模拟了一个公共交通系统，如地铁。问题是找到平面上两个顶点x和y之间的最短路径，其中R2 \ C上的距离度量由曼哈顿度量给出。此外，人们可以在C的任何点进入和离开公共交通，且没有延迟。假设公共交通以速度v > 1在C的边上移动。

Aichholzer et al. [AAP04] define by QC(x, y) the temporal distance according to the quickest path from x to y. This distance measure QC induces a Voronoi diagram on a set S of n vertices, which is called the city Voronoi diagram VC(S) of S w. r. t. C. Aichholzer et al. show that VC(S) is a subset of the additively and multiplicatively weighted[24] straight skeleton of an input which is based on S and C. This enables Aichholzer et al. [AAP04] to compute the city Voronoi diagram by using straight-skeleton algorithms. Interestingly, Aichholzer et al. were able to show that the input to the straight-skeleton algorithm has sufficiently nice properties such that the framework of abstract Voronoi diagrams of Klein [Kle89] can be applied. Note that for general input it is not possible to interpret the straight skeleton as abstract Voronoi diagram, see Section 2.1. This observation finally leads to an algorithm which uses O(n log n + c2 log c) time and O(n + c) optimal space.

Aichholzer等人[AAP04]根据从x到y的最快路径，将时间距离定义为QC(x, y)。这个距离度量QC在一个由n个顶点组成的集合S上诱导出一个Voronoi图，该图被称为city Voronoi diagram VC(S) of S w. r. t. C。Aichholzer等人表明，VC(S)是基于S和C的输入数据的加性和乘性加权[24]直线骨架的子集。这使得Aichholzer等人[AAP04]能够通过使用直线骨架算法来计算城市Voronoi图。有趣的是，Aichholzer等人能够证明直线骨架算法的输入具有足够好的性质，以至于可以应用Klein [Kle89]的抽象Voronoi图框架。请注意，对于一般输入，不可能将直线骨架解释为抽象Voronoi图，参见第2.1节。这一观察最终导致了一种算法，该算法使用O(n log n + c2 log c)时间和O(n + c)最优空间。

In order to make straight skeletons applicable in practice we seek an algorithm which is (i) easy to implement and (ii) exhibits an actual runtime that is relatively close to linear in the input size. Fortune started his introduction [For00] to the 27th volume of Algorithmica with the following words:

为了使直线骨架在实践中适用，我们寻求一种算法，该算法（i）易于实现，并且（ii）表现出与输入大小相对接近线性的实际运行时间。Fortune 在 Algorithmica 第 27 卷的介绍 [For00] 中以如下文字开头：

It is notoriously difficult to obtain a practical implementation of an abstractly described geometric algorithm. This difficulty arises in part from the conceptual complexity of many geometric algorithms, which often use sophisticated data structures or require other complex algorithms as subroutines. The difficulty also arises because many algorithms are designed and described to achieve good asymptotic behavior in the worst case, ignoring behavior in more realistic situations.

众所周知，要获得抽象描述的几何算法的实际实现是极其困难的。这种困难部分源于许多几何算法的概念复杂性，这些算法通常使用复杂的数据结构或需要其他复杂算法作为子程序。这种困难也源于许多算法的设计和描述是为了在最坏情况下实现良好的渐近行为，而忽略了在更实际情况下的行为。

In this chapter we present a novel straight-skeleton algorithm which is (i) easy to implement, (ii) exhibits a runtime that is close to linear in practice, (iii) accepts planar straight-line graphs as input, and (iv) has a lower worst-case runtime complexity than the triangulationbased algorithm by Aichholzer and Aurenhammer [AA98].

在本章中，我们提出了一种新颖的直线骨架算法，该算法 (i) 易于实现，(ii) 在实践中表现出接近线性的运行时间，(iii) 接受平面直线图作为输入，并且 (iv) 具有比 Aichholzer 和 Aurenhammer [AA98] 基于三角剖分的算法更低的worst-case运行时间复杂度。

We start with an investigation of the number of flip events that occur for the algorithm by Aichholzer and Aurenhammer [AA98] in Section 2.2. We first show a few results regarding the gap between Ω(n2) and O(n3) for the worst-case number of flip events and prove that we can exploit Steiner triangulations in order to completely avoid all flip events. This insight motivates an algorithm for straight skeletons of non-degenerate polygons that is based on motorcycle graphs, see Section 2.3. In order to generalize this algorithm to arbitrary planar straight-line graphs we introduce a generalization of the motorcycle graph in Section 2.4 and present an extension of our straight-skeleton algorithm in Section 2.5. Extensive runtime experiments in Section 2.5.4 reveal that our straight-skeleton implementation Bone exhibits an actual runtime consumption of O(n log n) in practice. Most results in the Sections 2.2 to 2.5 have been presented in the following publications:

我们首先在第2.2节中研究Aichholzer和Aurenhammer [AA98]算法中发生的翻转事件的数量。我们首先展示一些关于最坏情况下翻转事件数量的Ω(n2)和O(n3)之间差距的结果，并证明我们可以利用Steiner三角剖分来完全避免所有翻转事件。这一见解促使我们提出了一种基于摩托车图的非退化多边形直线骨架算法，见第2.3节。为了将该算法推广到任意平面

直线图，我们在第2.4节中引入了摩托车图的推广，并在第2.5节中提出了直线骨架算法的扩展。第2.5.4节中大量的运行时实验表明，我们的直线骨架实现Bone在实践中表现出O(n log n)的实际运行时消耗。第2.2节至2.5节中的大多数结果已在以下出版物中发表：
• [HH10b] S. Huber and M. Held. Straight Skeletons and their Relation to Triangulations. In Proc. 26th Europ. Workshop Comput. Geom., pages 189–192, Dortmund, Germany, Mar 2010

• [HH10b] S. Huber 和 M. Held. Straight Skeletons and their Relation to Triangulations. In Proc. 26th Europ. Workshop Comput. Geom., pages 189–192, Dortmund, Germany, Mar 2010

• [HH10a] S. Huber and M. Held. Computing Straight Skeletons of Planar Straight-Line Graphs Based on Motorcycle Graphs. In Proc. 22nd Canad. Conf. Comput. Geom. (CCCG 2010), pages 187–190, Winnipeg, Canada, Aug 2010

• [HH10a] S. Huber 和 M. Held. 基于摩托车图计算平面直线图的直线骨架. 发表于第22届加拿大计算几何会议 (CCCG 2010), 第187–190页, 加拿大温尼伯, 2010年8月

• [HH11c] S. Huber and M. Held. Theoretical and Practical Results on Straight Skeletons of Planar Straight-Line Graphs. In Proc. 27th Annu. ACM Sympos. Comput. Geom., Paris, France, to be published 2011

• [HH11c] S. Huber 和 M. Held. 关于平面直线图的直线骨架的理论和实践结果。发表于第27届ACM计算几何学年会，法国巴黎，待出版，2011
2.1 g e o m e t r i c p r o p e r t i e s o f t h e s t r a i g h t s k e l e t o n

2.1 直线骨架的几何属性

In this section we build a collection of geometric properties of straight skeletons and their relation to motorcycle graphs. Most of the following geometric properties were presented in [AAAG95, AA98, EE99, CV07]. Throughout this section, we denote by G a planar straightline graph. It turns out that the terrain T (G) is often useful to prove various properties of the straight skeleton and its relation to motorcycle graphs. In many cases the proofs become much easier when the scene considered is lifted to R3. The following lemma is a simple but useful tool for proofs of this kind.

在本节中，我们构建了直线骨架的几何属性集合，以及它们与摩托车图的关系。以下大多数几何属性在[AAAG95, AA98, EE99, CV07]中提出。在本节中，我们用G表示一个平面直线图。事实证明，地形T(G)通常有助于证明直线骨架的各种属性及其与摩托车图的关系。在许多情况下，当所考虑的场景被提升到R3时，证明会变得更容易。以下引理是一个简单但有用的工具，可用于此类证明。

Lemma 2.1. Consider two distinct points p, q on T (G). Then the straight line through p and q has a slope of at most 1.

引理 2.1. 考虑 T(G) 上的两个不同的点 p, q。则通过 p 和 q 的直线斜率至多为 1。

Proof. We project p and q onto the plane R2 × {0} and denote the results by p′ and q′. The intersection of T (G) with a vertical curtain above [p′ q′] results in a plane polygonal chain that starts at p and ends at q. All segments of this chain have a slope of at most 1, because each segment results from the intersection of the vertical curtain with a facet of T (G) and all facets have exactly slope 1. As a consequence, the supporting line pq through p and q has also a slope of at most 1.

证明。我们将p和q投影到平面R2 × {0}上，并将结果表示为p′和q′。T (G)与[p′ q′]上方垂直幕帘的交集产生一个平面多边形链，该链从p开始，到q结束。这条链的所有线段的斜率最多为1，因为每个线段都是垂直幕帘与T (G)的一个面的交集的结果，并且所有面的斜率都恰好为1。因此，通过p和q的支撑线pq的斜率也最多为1。

Definition 2.2. We denote by e(t) the set of points that are occupied by the wavefront edge e at time t and by e(t) we denote the supporting line of e(t). If e is emanated by a terminal vertex or an isolated vertex v of G then we define e(0) := limt↘0 e(t).

定义 2.2。我们用 e(t) 表示在时间 t 时，波前边缘 e 所占据的点的集合，用 e(t) 表示 e(t) 的支撑线。如果 e 是由 G 的一个终端顶点或孤立顶点 v 发出的，那么我们定义 e(0) := limt↘0 e(t)。

Note that the propagating wavefront edge e can be split at certain points in time. That is, e(t) can be expressed as the union of straight-line segments. Furthermore, we get that f (e) = St≥0 e(t). For a terminal vertex v we obtain that e(0) is equal to the supporting line of v that is perpendicular to the single incident edge of v.

注意，传播波前边缘 e 可以在某些时间点分裂。也就是说，e(t) 可以表示为直线段的并集。此外，我们得到 f (e) = St≥0 e(t)。对于终端顶点 v，我们得到 e(0) 等于 v 的支撑线，该支撑线垂直于 v 的单条入射边。

Aichholzer et al. [AAAG95] proved the following lemma for general bisector graphs of simple polygons instead of straight skeletons. These graphs correspond to generalized roofs as discussed in Section 1.2.3. Aichholzer and Aurenhammer [AA96] presented this lemma for planar straight-line graphs G. We rephrase their original proof using Lemma 2.1.

Aichholzer等人[AAAG95]证明了以下引理，该引理适用于简单多边形的一般二分线图，而不是直线骨架。这些图对应于第1.2.3节中讨论的广义屋顶。Aichholzer和Aurenhammer [AA96]提出了针对平面直线图G的此引理。我们使用引理2.1改写了他们的原始证明。

Lemma 2.3 ([AA96]). Let e be a wavefront edge of G. The face f (e) is monotone w. r. t. e(0).

引理 2.3 ([AA96]).设 e 是 G 的一个波前边缘。面 f (e) 关于 e(0) 是单调的。

Proof. It has to be shown that an intersection of f (e) with a line perpendicular to e(0) is connected. Assume that this is not the case for a line l that is perpendicular to e(0). One can find two points x and y on the boundary of f (e) such that the open segment (x, y) is

not contained in f (e). We lift the problem to R3 and put a vertical curtain on [xy], intersect it with T (G) and obtain a polygonal chain L. Let us denote its endpoints with xˆ and yˆ, which are contained in fˇ(e). The supporting line of xˆ, yˆ has a slope of exactly 1, but note that L is not contained in fˇ(e) by assumption. Hence, L contains at least one segment which has a slope that is greater than 1. But this is a contradiction to Lemma 2.1.

证明。必须证明f(e)与垂直于e(0)的直线的交集是连通的。假设对于垂直于e(0)的直线l，情况并非如此。可以找到f(e)边界上的两个点x和y，使得开线段(x, y)不包含在f(e)中。我们将问题提升到R3，并在[xy]上放置一个垂直幕帘，将其与T(G)相交，得到一条折线L。我们用xˆ和yˆ表示其端点，它们包含在fˇ(e)中。xˆ, yˆ的支撑线的斜率恰好为1，但请注意，根据假设，L不包含在fˇ(e)中。因此，L至少包含一个斜率大于1的线段。但这与引理2.1相矛盾。
Lemma 2.4 ([AA96]). The straight skeleton S (G) has exactly 2n + t − 2 number of nodes, where t denotes the number of terminals in G.

引理 2.4 ([AA96]).直线骨架 S (G) 恰好有 2n + t − 2 个节点，其中 t 表示 G 中的终端数量。
Similar to Lemma 1.2, the lemma holds if the degree of all nodes is three. In the presence of multi split events the degree of the resulting nodes have to be taken into account accordingly.

类似于引理 1.2，如果所有节点的度数为 3，则该引理成立。在存在多重分裂事件的情况下，必须相应地考虑所得节点的度数。

The proof of Aichholzer and Aurenhammer is based on the analysis of their algorithm, see Section 1.4.2.2. Recall that their algorithm computes a constrained triangulation of G and then simulates the wavefront by maintaining the triangulation. A topological event of the wavefront corresponds to a decrease of the number of triangles by one. Note that the so-called flip events leave the number of triangles invariant. Therefore, the number of events happening to the wavefront corresponds to the number of triangles collapsed. The number of triangles that remain after the last event happened corresponds to the number of unbounded arcs resp. infinite nodes of S (G). Aichholzer and Aurenhammer [AA96] argue that the number of initial triangles equals the number of nodes, including the infinite nodes.

Aichholzer和Aurenhammer的证明基于对其算法的分析，参见第1.4.2.2节。回想一下，他们的算法计算G的约束三角剖分，然后通过维护三角剖分来模拟波前。波前的拓扑事件对应于三角形数量减少一个。请注意，所谓的翻转事件使三角形的数量保持不变。因此，波前发生的事件数对应于塌陷的三角形数。最后一个事件发生后剩余的三角形数量对应于无界弧的数量，即S (G)的无限节点。S (G)的无限节点。Aichholzer和Aurenhammer [AA96]认为，初始三角形的数量等于节点的数量，包括无限节点。

Proof. One observes that the initial triangulation tessellates the plane into 2n − 2 triangles. Let us consider an embedding of G onto the three-dimensional sphere, in a local neighborhood of the north pole. We identify the north pole with the origin of the plane and we consider the south pole as an additional vertex which models the locus at infinity. Then an induction-type argument easily shows that any triangulation of n ≥ 3

vertices plus the infinite vertex comprises 2n − 2 triangles. Finally, each terminal vertex gives rise to an additional triangle in order to take the additional wavefront edges at terminal vertices into account. Summarizing, the initial triangulation comprises 2n − 2 + t triangles.

证明。可以观察到，初始三角剖分将平面分割成 2n − 2 个三角形。让我们考虑将 G 嵌入到三维球面上，在北极的局部邻域内。我们将北极识别为平面的原点，并将南极视为一个额外的顶点，它模拟无穷远处的轨迹。然后，一个归纳类型的论证很容易表明，n ≥ 3 个顶点加上无穷远顶点的任何三角剖分都包含 2n − 2 个三角形。最后，每个终端顶点都会产生一个额外的三角形，以便将终端顶点处的额外波前边缘考虑在内。总结一下，初始三角剖分包含 2n − 2 + t 个三角形。
the straight skeleton as a voronoi diagram An obvious yet essential question is whether the straight skeleton can be interpreted as a generalized or abstract Voronoi diagram. The benefit of such a connection is obvious: Voronoi diagrams have been extensively studied in the past decades and there are efficient algorithms for a wide variety of generalizations of Voronoi diagrams [Kle89, KMM93, Yap87, AS95]. Aichholzer and Aurenhammer [AA96] considered the abstract Voronoi framework by Klein [Kle89] in order to investigate this question. The idea of Klein is that one does not consider a distance function in order to define the Voronoi region of each input site, but to define mutual bisector curves between each pair of input sites. Klein [Kle89] gave a set of axioms that need be fulfilled by this set of bisectors, in order to define an abstract Voronoi diagram. The generalized Voronoi diagrams of points, straight-line segments and circular arcs are covered by this framework as well. In addition, generalizations to other distance functions can be represented within this framework. Recently, Klein et al. [KLN09] revisited this framework and presented a conciser set of axioms for the bisector system.

将直骨架视为Voronoi图：一个显而易见但至关重要的问题是，直骨架是否可以被解释为广义或抽象的Voronoi图。这种联系的益处是显而易见的：Voronoi图在过去的几十年里得到了广泛的研究，并且对于各种Voronoi图的推广，存在着高效的算法[Kle89, KMM93, Yap87, AS95]。Aichholzer和Aurenhammer [AA96] 考虑了Klein [Kle89] 的抽象Voronoi框架，以便研究这个问题。Klein的思想是不考虑距离函数来定义每个输入站点的Voronoi区域，而是定义每对输入站点之间的相互平分曲线。Klein [Kle89] 给出了一组需要被这组平分线满足的公理，以便定义一个抽象的Voronoi图。点、直线段和圆弧的广义Voronoi图也包含在这个框架中。此外，对其他距离函数的推广也可以在这个框架内表示。最近，Klein 等人 [KLN09] 重新审视了这个框架，并为平分线系统提出了一套更简洁的公理。
Unfortunately, Aichholzer and Aurenhammer [AA96] pointed out that the framework of Klein cannot be applied to define straight skeletons. Let us consider a bisector between two input sites a, b motivated by the straight skeleton. That is, the bisector is the set of points that are reached at the same time by the wavefronts of a and b. The claim is that the resulting bisector system does not fulfill the axioms of [KLN09]. Following the notation of [KLN09] we denote by J(a, b) the bisector between the site a and b. The bisector tessellates the plane into two halves. The half which belongs to a is denoted by D(a, b) and the other by D(b, a). The Voronoi region V (a) of a is defined by the intersection Ts□=a D(a, s) among all input sites s. See Figure 18 for an example. We observe that the Voronoi region of e1 is influenced by the presence of e3. However, the straight-skeleton

wavefronts of e3 are blocked by e2. Note that if we would remove e2 then J(e1, e3) would pose a correct part of the resulting straight skeleton. The basic reason for this strange behavior is that the bisectors motivated by straight skeletons do not fulfill the axioms of Klein et alii. In particular, their axioms include that D(e1, e2) ∩ D(e2, e3) ⊂ D(e1, e3), which is clearly not the case in our example: The point p is contained in the set at the left-hand side but not in the right-hand side. Moreover, p is not contained in any Voronoi region at all.

不幸的是，Aichholzer和Aurenhammer [AA96] 指出，Klein的框架不能应用于定义直线骨架。让我们考虑由直线骨架驱动的两个输入点 a, b 之间的平分线。也就是说，平分线是由 a 和 b 的波前同时到达的点的集合。该论断是，由此产生的平分线系统不满足 [KLN09] 的公理。按照 [KLN09] 的符号，我们用 J(a, b) 表示点 a 和 b 之间的平分线。该平分线将平面分割成两个半平面。属于 a 的半平面用 D(a, b) 表示，另一个用 D(b, a) 表示。a 的 Voronoi 区域 V (a) 定义为所有输入点 s 之间的交集 Ts□=a D(a, s)。参见图 18 的示例。我们观察到 e1 的 Voronoi 区域受到 e3 存在的影响。然而，e3 的直线骨架波前被 e2 阻挡。请注意，如果我们移除 e2，那么 J(e1, e3) 将构成所得直线骨架的正确部分。这种奇怪行为的基本原因是，由直线骨架驱动的平分线不满足 Klein 等人的公理。特别是，他们的公理包括 D(e1, e2) ∩ D(e2, e3) ⊂ D(e1, e3)，这在我们的例子中显然不是这种情况：点 p 包含在左侧的集合中，但不包含在右侧的集合中。此外，p 根本不包含在任何 Voronoi 区域中。

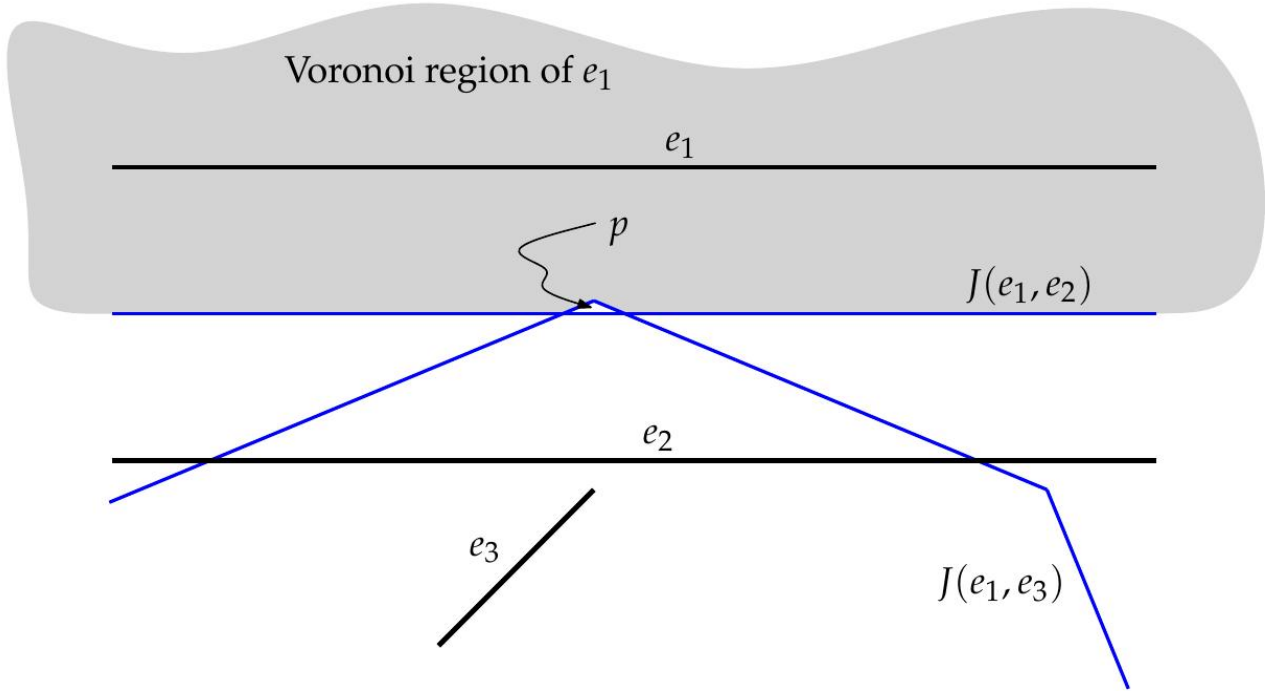


Figure 18: An attempt to define S(G) using abstract Voronoi diagrams. The bisectors J(e1, e2) and J(e1, e3) are depicted in blue. The attempt fails because e3 influences the Voronoi region of e1. In particular, the point p is not contained in the Voronoi region of e1.

图 18：尝试使用抽象 Voronoï 图定义 S(G)。平分线 J(e1, e2) 和 J(e1, e3) 以蓝色描绘。该尝试失败，因为 e3 影响了 e1 的 Voronoï 区域。特别地，点 p 不包含在 e1 的 Voronoï 区域中。

The concept of abstract Voronoi diagrams by Klein et al. [Kle89, KLN09] distills the very essence of Voronoi diagrams from an abstract point of view. Having this interpretation in mind, straight skeletons are essentially different to Voronoi diagrams, even though they share common properties at the first sight. Nevertheless, Aichholzer and Aurenhammer [AA98] mentioned that for special case of rectilinear polygons P, the Voronoi diagram of P in the L∞-space and the straight skeleton of P coincide.

Klein等人[Kle89, KLN09]提出的抽象Voronoi图的概念，从抽象的角度提炼了Voronoi图的本质。考虑到这种解释，尽管直线骨架乍一看与Voronoi图有共同的性质，但它们本质上是不同的。然而，Aichholzer和Aurenhammer [AA98]提到，对于直线多边形的特殊情况P，P在L∞-space中的Voronoi图与P的直线骨架重合。

the terrain model and piecewise-linear functions Aichholzer et al. [AAAG95] investigated whether partially defined linear distance functions could be used in order to obtain an alternative, non-procedural way of defining the straight skeleton S (G). More precisely, the idea is to define for each wavefront edge e a real-valued linear function de(.) on a subset of R2 such that the lower envelope of the graphs of these functions forms the terrain T (G). For a point $p \in R^2$, the value de(p) would indicate the time when p is hit by the wavefront edge e. Aichholzer et al. already showed that the domain of de cannot only depend on e: In Figure 18, the domain of de3 must somehow take care for the presence of e2. We can summarize this insight as follows:

Aichholzer等人[AAAG95]研究了地形模型和分段线性函数，探讨是否可以使用部分定义的线性距离函数来获得定义直线骨架S(G)的另一种非过程化方法。更准确地说，其思想是为每个波前边缘e定义一个实值线性函数de(.)，该函数定义在R2的子集上，使得这些函数图的下包络形成地形T(G)。对于点$p \in R^2$，值de(p)将指示p被波前边缘e击中的时间。Aichholzer等人已经表明，de的定义域不能仅依赖于e：在图18中，de3的定义域必须以某种方式考虑e2的存在。我们可以将这一见解总结如下：
Observation 2.5 ([AAAG95]). If one attempts to define a partially linear function for each wavefront edge e of G such that the lower envelope of their graphs is identical to T (G) then the domain of each function de cannot only depend on the defining wavefront edge e.

观察 2.5 ([AAAG95]).如果尝试为G的每个波前边缘e定义一个部分线性函数，使得它们的图形的下包络线与 T (G) 相同，那么每个函数de的域不能仅依赖于定义的波前边缘e。

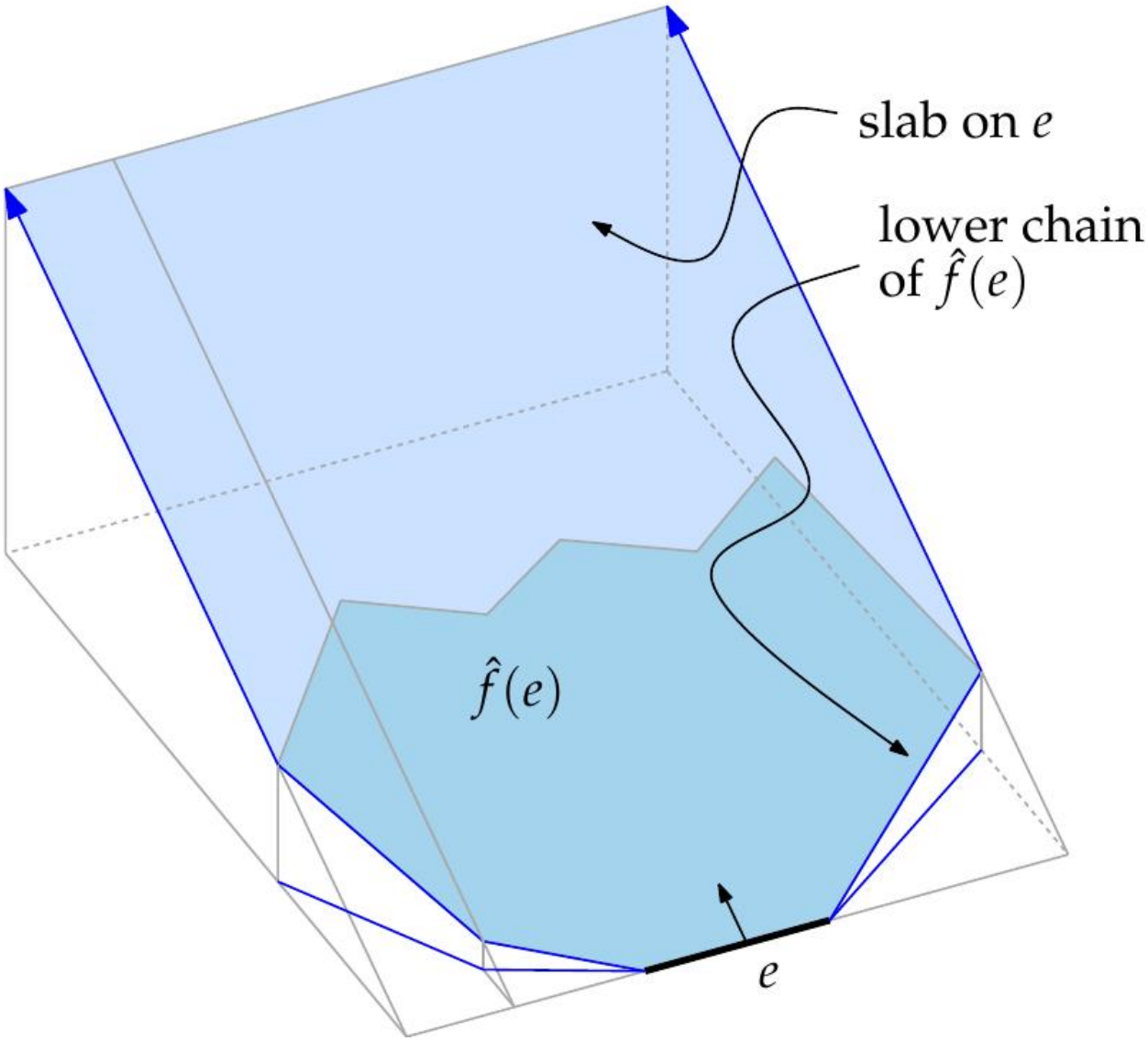


Figure 19: The slabs in R3 are bounded from below by the lower chain of fˆ(e).

图 19：R3中的板由fˆ(e)的下链从下方限定。

Let us consider the face f (e) of a wavefront edge e of G. By Lemma 2.3, the boundary of f (e) consists of two monotonic chains with respect to e(0).

我们考虑G的波前边缘e的面f (e)。根据引理2.3，f (e)的边界由关于e(0)的两条单调链组成。

Definition 2.6 (lower/upper chain of a face). The lower chain and the upper chain of f (e) are the two monotonic chains w. r. t. e(0) that constitute the boundary of f (e). The lower chain contains e(0).

定义 2.6（面的下链/上链）[NT0]。f (e) 的下链和上链是关于 e(0) 的两个单调链，它们构成了 f (e) 的边界。下链包含 e(0)。

For the lower chain C of a face f (e), we denote by Cˆ the projection of C onto T (G). For each wavefront edge e we define an infinite slab that lies on the supporting plane of fˆ(e) and is bounded from below by Cˆ and by two rays at both endpoints of Cˆ that are

perpendicular to e(0), see Figure 19. Eppstein and Erickson [EE99] proved the following theorem.

对于面f(e)的下链C，我们用Cˆ表示C在T(G)上的投影。对于每个波前边缘e，我们定义一个无限板，它位于fˆ(e)的支撑平面上，并由Cˆ和Cˆ两个端点处的两条垂直于e(0)的射线从下方限定，参见图19。Eppstein和Erickson [EE99] 证明了以下定理。

Theorem 2.7 ([EE99]). The lower envelope of all slabs of the wavefront edges of G is identical to T (G).

定理 2.7 ([EE99])[NT0]。G的波前边缘的所有板的下包络线与 T (G) 相同。

Proof. We first note that for any wavefront edge e of G, the lifted face fˆ(e) is contained in the slab that we defined for e. Hence, it remains to show that the terrain T (G) is not above the lower envelope of the slabs at any point.

证明。我们首先注意到，对于G的任何波前边e，提升的面fˆ(e)包含在我们为e定义的板中。因此，剩下的就是证明地形T(G)在任何点上都不高于板的下包络线。

Assume that p is a point on T (G) that indeed lies above the lower envelope. Hence, we can project p vertically onto the lower envelope and obtain a point p′. Then we can project p′ along the steepest descent of the slab, where p′ lies on, until we hit a point q. Clearly, q and p are on T (G). Note that qp′ has slope 1 and, thereby, qp must have slope greater than 1. This is a contradiction to Lemma 2.1.

假设p是T(G)上的一个点，并且确实位于下包络线的上方。因此，我们可以将p垂直投影到下包络线上，得到一个点p′。然后，我们可以沿着平板最陡的下降方向投影p′，其中p′位于该平板上，直到我们到达一个点q。显然，q和p都在T(G)上。请注意，qp′的斜率为1，因此，qp的斜率必须大于1。这与引理2.1相矛盾。

In the original work by Eppstein and Erickson [EE99] the theorem above is presented for polygonal input, but it is claimed that it also extends to planar straight-line graphs. Further, they actually used the following set of slabs: For every edge e they defined an edge slab, which is bounded from below by e and by two perpendicular rays at each endpoint of e. For every reflex vertex v of G they defined two reflex slabs each of which is bounded from below by the valley incident to v and, for each, two rays that are perpendicular to the two incident edges of v. Strictly speaking, in their original phrasing they did not consider that vertex events could happen, i. e., that a valley of T (G) is not incident to a vertex of G. Nevertheless, the basic idea of their proof also works in the general case, which is based on a careful consideration of the relative positions of the moving wavefront edges. We presented a proof that is based on Lemma 2.1 instead. Our proof appears to be simpler and immediately applies to arbitrary planar straight-line graphs G.

在Eppstein和Erickson的原始著作[EE99]中，上述定理是针对多边形输入提出的，但他们声称该定理也适用于平面直线图。此外，他们实际上使用了以下slab集合：对于每条边e，他们定义了一个边slab，其下界为e，上界为e的每个端点处的两条垂直射线。对于G的每个凹顶点v，他们定义了两个凹slab，每个凹slab的下界为与v相邻的谷，并且对于每个凹slab，有两条与v的两条相邻边垂直的射线。严格来说，在他们最初的措辞中，他们没有考虑到顶

点事件可能发生，即T(G)的谷不与G的顶点相邻。然而，他们的证明的基本思想也适用于一般情况，这是基于对移动波前边缘的相对位置的仔细考虑。我们提出了一个基于引理2.1的证明。我们的证明似乎更简单，并且可以直接应用于任意平面直线图G。
As Eppstein and Erickson [EE99] mentioned, Theorem 2.7 leads to a nice alternative interpretation of the straight skeleton S (G), based on a partially defined linear functions. The following corollary formulates this interpretation, where d denotes the infimum distance, that is d(A, B) := infx∈A,y∈B d(x, y) for two point sets A, B ⊂ R^2.

正如Eppstein和Erickson [EE99] 所提到的，定理2.7引出了对直线骨架S(G)的一个很好的替代解释，基于一个部分定义的线性函数。以下推论阐述了这个解释，其中d表示下确界距离，即对于两个点集A, B ⊂ R^2，d(A, B) := infx∈A,y∈B d(x, y)。

Corollary 2.8. Let e and e′ denote two wavefront edges. For any point p ∈ f (e), whose orthogonal projection line onto e′(0) intersects the lower chain of f (e′), holds d(p, e(0)) ≤ d(p, e′(0)).

推论 2.8. 设 e 和 e′ 表示两个波前边缘。对于任何点 p ∈ f (e)，其在 e′(0) 上的正交投影线与 f (e′) 的下链相交，则 d(p, e(0)) ≤ d(p, e′(0)) 成立。

This result enables us to define a point set that is similar to the cone of influence in the theory of Voronoi diagrams, cf. [Hel91]. That is, for each wavefront edge e we define a set CI (e) by the projection of the slab of e onto the plane. Next, we define a distance function de between a wavefront edge e and a point p by

这个结果使我们能够定义一个点集，它类似于Voronoi图理论中的影响锥，参见[Hel91]。也就是说，对于每个波前边缘e，我们通过将e的板投影到平面上来定义一个集合CI (e)。接下来，我们通过以下方式定义波前边缘e和点p之间的距离函数de

$$d_e(p) := (d$$

(
e
(
0
)
,
p
)
if
p
∈
C
I
(
e
)
∞
o
t
her
wise.
(2.1)
Using these distance functions de, we can rephrase the previous corollary to

使用这些距离函数 de，我们可以将之前的推论改述为

f
(

e

)

=

\

e

′

∈

E

{

p

∈

R

2

:

d

e

(

p

)

≤

d

e

′

(

p

)

}

,

(2.2)
where E denotes the set of wavefront edges. At the first sight, this result seems to be an alternative characterization of S (G). However, we want to remark that the sets CI (e) depend on the length of the reflex arcs of S (G). In other words, the representation of the faces via (2.2) does not serve as an alternative characterization of straight skeletons.

其中 E 表示波前边的集合。乍一看，这个结果似乎是 S (G) 的另一种表征。然而，我们想指出的是，集合 CI (e) 取决于 S (G) 的反射弧的长度。换句话说，通过 (2.2) 表示面并不能作为直线骨架的另一种表征。

Eppstein and Erickson [EE99] remarked that Corollary 2.8 does not pose a contradiction to Observation 2.5: The domain of the distance function de depends on the length of certain reflex arcs and, therefore, does not only depend on the wavefront edge e.

Eppstein和Erickson [EE99] 指出，推论2.8与观察2.5并不矛盾：距离函数de的定义域取决于某些反射弧的长度，因此不仅仅取决于波前边缘e。

Lemma 2.9. The lower chain of the face f (e) of a wavefront edge e is convex.

引理 2.9. 波前边缘 e 的面 f (e) 的下链是凸的。

Proof. The lemma asserts that the interior angle at each vertex of the lower chain C of f (e) is at most 180°. The segment e(0) encloses with its incident arcs an angle less than 180°, because these arcs lie on the bisectors of e and adjacent edges of e. Hence, it suffices to show that the interior angle at any straight skeleton node on C is at most 180°. We assume the contrary: Suppose that there is a node v on C whose interior angle is reflex. Without loss of generality, we may assume that v is (i) closest to e(0) and (ii) on the left part of C w. r. t. e(0), see Figure 20.

证明。该引理断言，f (e) 的下链 C 的每个顶点的内角至多为 180°。线段 e(0) 与其入射弧所围成的角度小于 180°，因为这些弧位于 e 和 e 的相邻边的平分线上。因此，只需证明 C 上任何直线骨架节点的内角至多为 180°。我们假设相反的情况：假设在 C 上存在一个节点 v，其内角为反角。不失一般性，我们可以假设 v (i) 最接近 e(0)，并且 (ii) 位于 C 的左侧，相对于 e(0)，参见图 20。

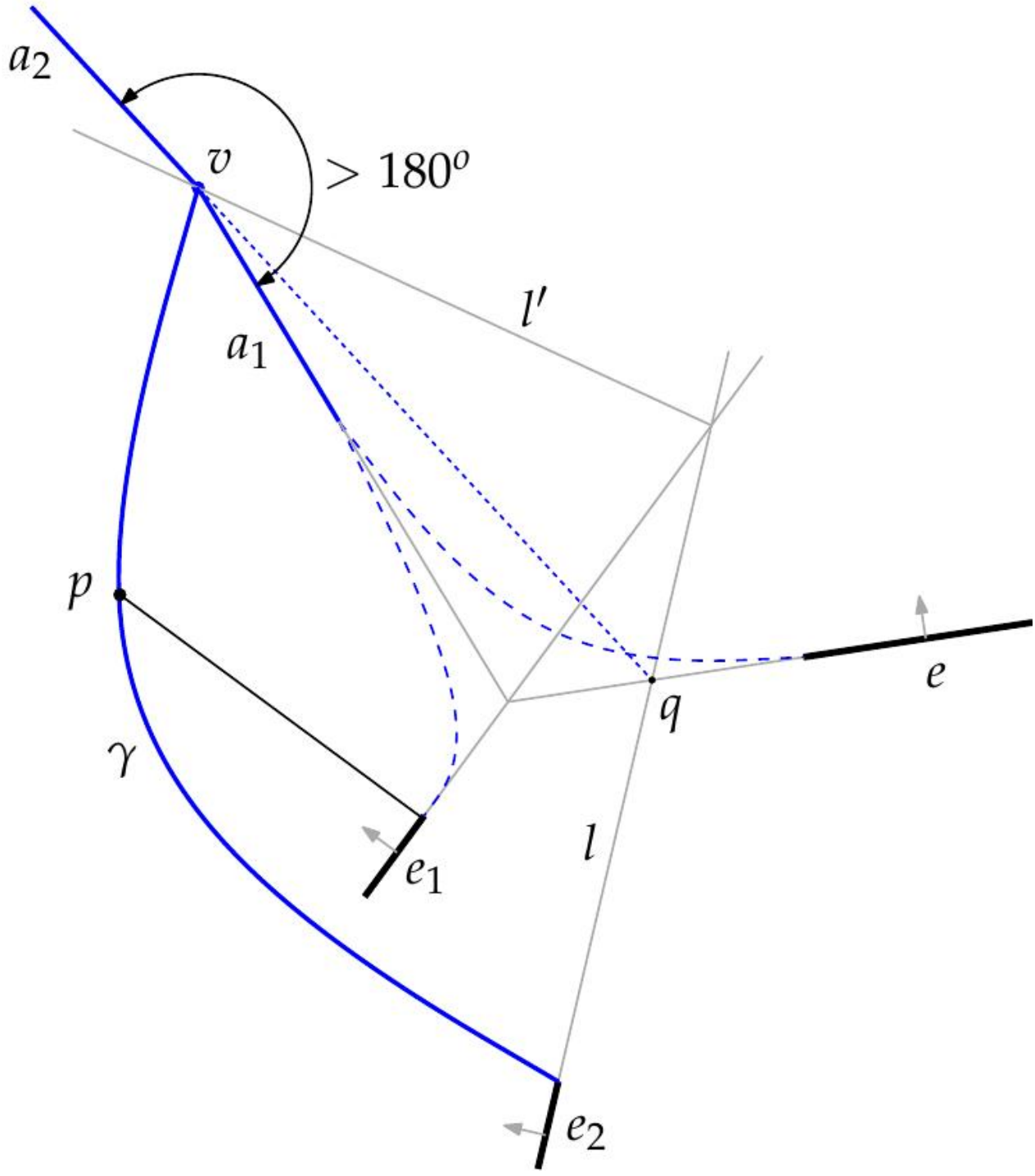


Figure 20: The vertex v on the lower chain of f (e) cannot be reflex.

图 20：f (e) 的下链上的顶点 v 不可能是反身的。

We denote by a1 and a2 the arcs on C that are incident to v such that the sub-chain from a1 to e(0) is convex. Further, we denote by e1 and e2 the other two wavefront edges whose faces contain the arcs a1 and a2, respectively. The supporting line of a2 intersects e(0) at a point q. There exists a supporting line l of q such that a2 lies on the bisector of l and e. Hence, the edge e2 lies on l, by construction. Further, we note that e2(0) lies behind e(0) and also behind e1(0) w. r. t. the corresponding propagation direction.

我们用a1和a2表示C上与v关联的弧，使得从a1到e(0)的子链是凸的。此外，我们用e1和e2表示另外两条波前边缘，它们的面上分别包含弧a1和a2。a2的支撑线与e(0)相交于点q。存在q的一条支撑线l，使得a2位于l和e的平分线上。因此，根据构造，边缘e2位于l上。此外，我们注意到，关于相应的传播方向，e2(0)位于e(0)之后，也位于e1(0)之后。
Since e2 reaches v after some time, the lower chain of f (e2) contains a polygonal chain γ that connects e2(0) and v. We note that γ is monotone w. r. t. e2 by Lemma 2.3. Furthermore, γ is monotone w. r. t. the propagation direction of e2 because the points on γ are swept by the propagating edge e2. The curve γ contains a point p that can be projected orthogonally onto e1(0). Corollary 2.8 implies that d(p, e2(0)) ≤ d(p, e1(0)) because p ∈ f (e2). Let us denote by l′ the bisector between e1 and 2. That is, l′ consists of all points e1(t) ∩ e2(t), with t ≥ 0, and all points left to l′ are first reached e1. However, since p is left to l′ it follows that d(p, e1(0)) < d(p, e2(0)), which is a contradiction.

由于e2经过一段时间后到达v，因此f (e2)的下链包含连接e2(0)和v的折线γ。我们注意到，根据引理2.3，γ关于e2是单调的。此外，由于γ上的点被传播边e2扫过，因此γ关于e2的传播方向是单调的。曲线γ包含一个可以正交投影到e1(0)上的点p。推论2.8表明d(p, e2(0)) ≤ d(p, e1(0))，因为p ∈ f (e2)。让我们用l′表示e1和2之间的平分线。也就是说，l′由所有t ≥ 0的点e1(t) ∩ e2(t)组成，并且所有位于l′左侧的点首先到达e1。然而，由于p位于l′的左侧，因此d(p, e1(0)) < d(p, e2(0))，这是一个矛盾。

Lemma 2.10. The lower chain of a face f (e) consists of e(0) and reflex straight-skeleton arcs.

引理 2.10. 面 f (e) 的下链由 e(0) 和反射直线骨架弧组成。

The following theorem was proved by Cheng and Vigneron [CV07]. It establishes an essential connection between the motorcycle graph and the straight skeleton.

以下定理由 Cheng 和 Vigneron [CV07] 证明。它建立了摩托车图和直线骨架之间的本质联系。

Theorem 2.11 ([CV07]). Let P denote a simple non-degenerate polygon P. The reflex arcs of S (P) are covered by M(P).

定理 2.11 ([CV07])[NT0]。设 P 表示一个简单的非退化多边形 P。S (P) 的反射弧被 M(P) 覆盖。

Proof. We slightly rephrase the proof in [CV07] in order to fit our setting. First, we declare that S (P) and M(P) are restricted to the interior of the polygon P rather than being defined on the whole plane.

证明。我们稍微改写了[CV07]中的证明，以适应我们的设置。首先，我们声明S(P)和M(P)被限制在多边形P的内部，而不是定义在整个平面上。

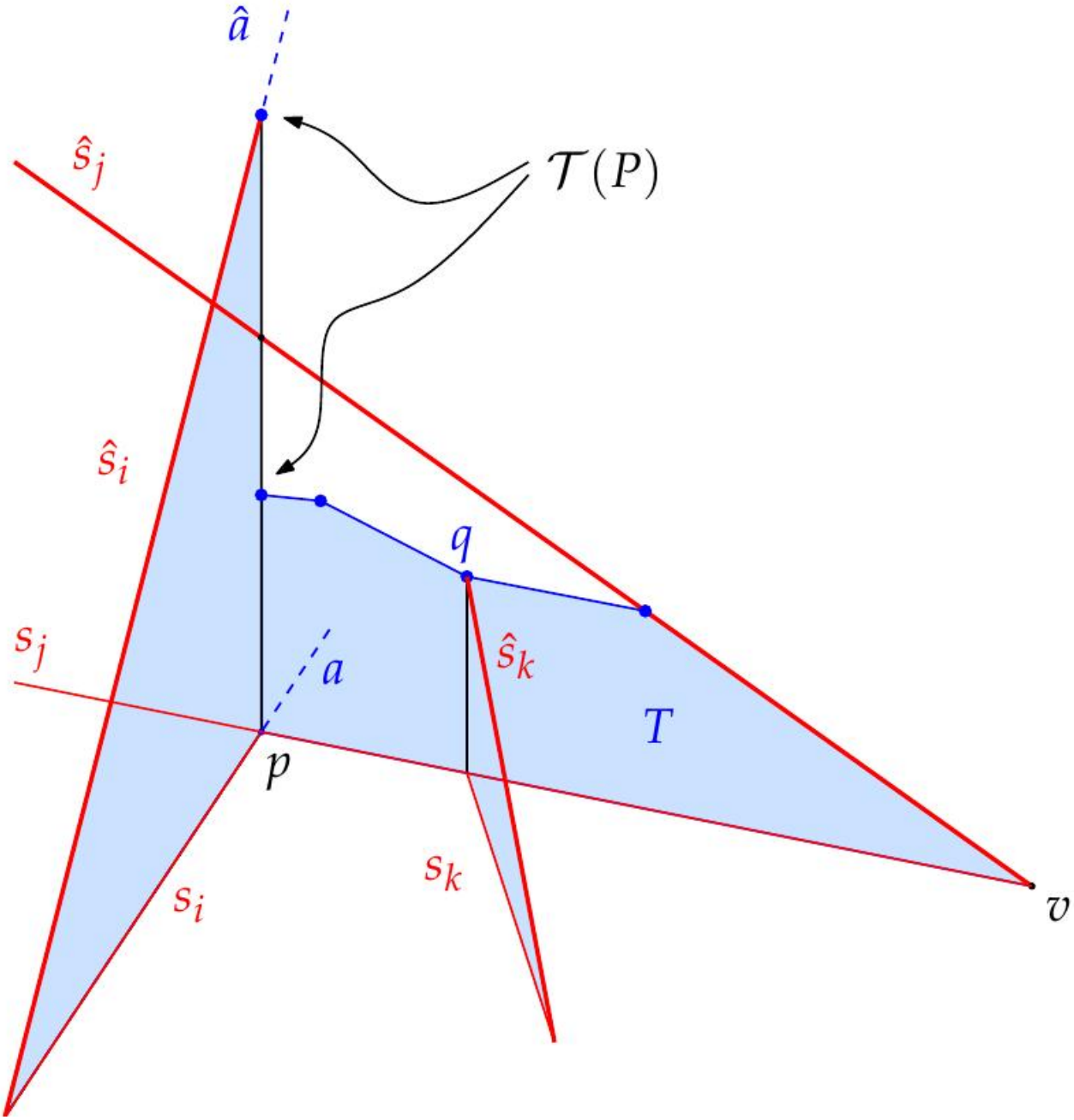


Figure 21: Reflex arcs are covered by motorcycle traces. The set T turns out to be convex which leads to a contradiction concerning the height of T (P) above p.

图 21：反射弧被摩托车轨迹覆盖。集合 T 结果是凸的，这导致了关于 T (P) 在 p 上方高度的矛盾。

Cheng and Vigneron lift the problem to R3 by means of the terrain model. Let m1, . . . , mr denote the motorcycles from M(P), where r denotes the number of reflex vertices of P. Further, let si denote the trace of the motorcycle mi and let sˆi denote the lifted trace of mˆ i by interpreting the third spatial dimension as the time. Recall that each motorcycle mi starts at a reflex wavefront vertex v of W (P, 0) and mi and v have the same velocity. Hence, the valley aˆ that belongs to v has the same inclination as the lifted motorcycle trace mˆ i.

Cheng和Vigneron借助地形模型将问题提升到R3。设m1, . . . , mr表示来自M(P)的摩托车，其中r表示P的反射顶点的数量。此外，设si表示摩托车mi的轨迹，设sˆi表示mˆ i的提升轨迹，通过将第三个空间维度解释为时间。回想一下，每辆摩托车mi都从W (P, 0)的反射波前顶点v开始，并且mi和v具有相同的速度。因此，属于v的谷aˆ与提升的摩托车轨迹mˆ i具有相同的倾斜度。
Now assume that the statement is false and there is indeed a reflex arc a of S (P) that is only partially covered by the motorcycle trace si. Since a continues on the track of mi, the motorcycle mi crashed into another motorcycle, say mj. Let p denote the intersection point of si and sj and let t∗ denote the height of T (P) above p. Among all reflex arcs that are only partially covered, a is chosen such that t∗ is lowest. Hence, up to the height t∗, all valleys of T (P) are covered by tilted motorcycle traces.

现在假设该陈述是错误的，并且确实存在一个S (P)的反射弧a，它仅被摩托车轨迹si部分覆盖。由于a继续在mi的轨道上，摩托车mi撞到了另一辆摩托车，比如mj。设p表示si和sj的交点，并设t∗表示T (P)在p上方的高度。在所有仅被部分覆盖的反射弧中，选择a使得t∗最低。因此，直到高度t∗，T (P)的所有谷都由倾斜的摩托车轨迹覆盖。

The trace sj starts at a reflex vertex v of P and contains p. Let T denote the intersection of a vertical curtain that is put on the segment [vp] with the set of points below T (P), see Figure 21. The claim is that T is convex. Assume, to the contrary, that T contains a reflex vertex on its top chain. Let q denote such a reflex vertex that is closest to v. The vertex q cannot be at height zero, which would mean that mj crashed into a wall — that is, an edge of P — at q. Note that T and sˆj start with the same inclination, which means that q is below or just at the same height as sˆj. Because q is a reflex vertex of T, there exists a valley at q, and because each valley is covered by a motorcycle trace until height t∗, there is a tilted motorcycle trace sˆk at q. So either sj crashed into sk or the valleys corresponding to mj and mk meet at q. Both cases are a contradiction to the initial assumptions. It follows that T is convex and hence below sˆj.

轨迹 sj 始于 P 的一个反射顶点 v，并且包含 p。设 T 表示垂直幕布（放置在 [vp] 线段上）与 T (P) 下方点集的交集，参见图 21。该论断是 T 是凸的。反之，假设 T 在其顶部链上包含一个反射顶点。设 q 表示这样一个离 v 最近的反射顶点。顶点 q 不可能在高度为零的位置，这意味着 mj 撞到了一堵墙——也就是 P 的一条边——在 q 处。请注意，T 和 sˆj 以相同的倾斜度开始，这意味着 q 低于或恰好与 sˆj 处于相同的高度。由于 q 是 T 的一个反射顶点，因此在 q 处存在一个谷，并且由于每个谷都被摩托车轨迹覆盖到高度 t∗，因此在 q 处存在一个倾斜的摩托车轨迹 sˆk。因此，要么 sj 撞到了 sk，要么对应于 mj 和 mk 的谷在 q 处相遇。这两种情况都与最初的假设相矛盾。由此可见，T 是凸的，因此位于 sˆj 的下方。

GEOMETRIC PROPERTIES OF THE STRAIGHT SKELETON

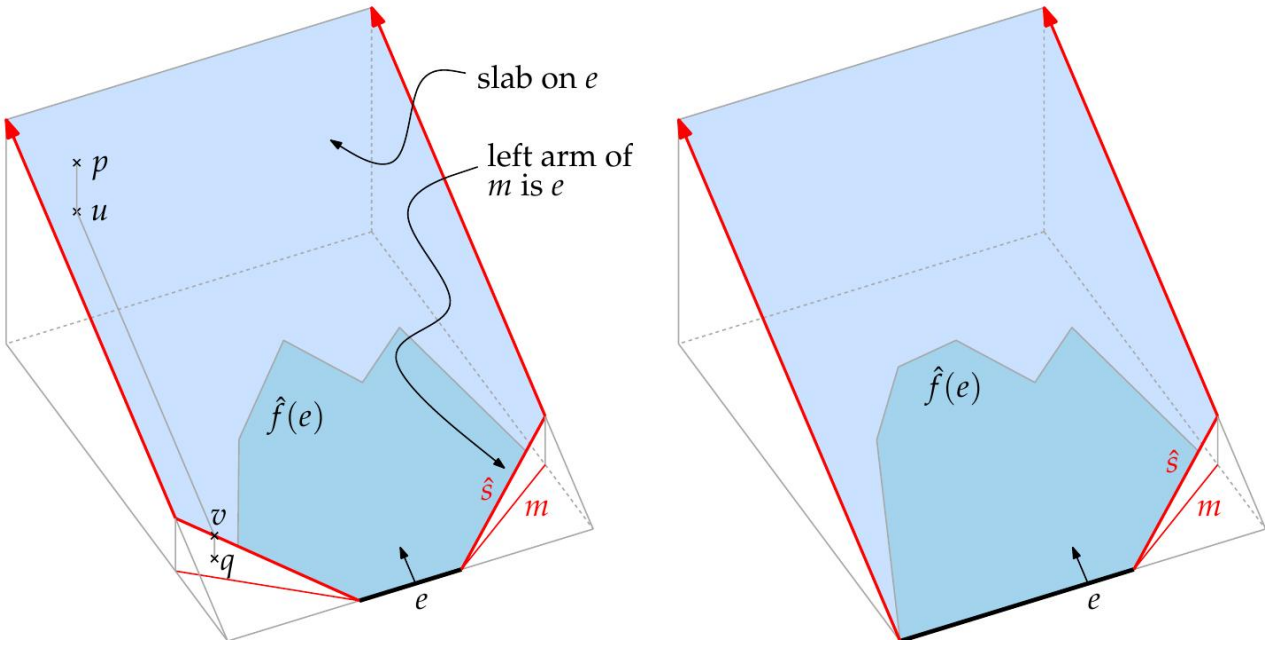


Figure 22: The slab at e is bounded from below by e and the tilted motorcycle traces of motorcycles that have e as an arm. Left: two motorcycles have e as an arm. Right: Only one motorcycle has e as an arm.

图 22：e 处的板由 e 以及以 e 为臂的倾斜摩托车轨迹从下方限定。左图：两辆摩托车以 e 为臂。右图：只有一辆摩托车以 e 为臂。

The height of T (P) at p is once given by the valley aˆ, on one hand, and by the height of T at p, on the other hand. But this is a contradiction, because aˆ is strictly above sˆj, which itself is above T.

一方面，p 处的 T (P) 的高度由谷 aˆ 给出；另一方面，由 p 处的 T 的高度给出。但这是一个矛盾，因为 aˆ 严格高于 sˆj，而 sˆj 本身又高于 T。

The following corollary is a byproduct of the previous proof.

以下推论是前一个证明的副产品。

Corollary 2.12 ([CV07]). Let P denote a simple non-degenerate polygon P. The tilted traces sˆ of M(P) are above or just at the same height as T (P).

推论 2.12 ([CV07]).设 P 表示一个简单的非退化多边形 P。M(P) 的倾斜轨迹 sˆ 位于 T (P) 之上或与其高度相同。

Let us revisit the slab construction from Theorem 2.7. Cheng and Vigneron [CV07] presented a different slab construction scheme that is based on the motorcycle graph. For the matter of consistency, we reformulate their construction as follows.

让我们回顾一下定理2.7中的平板构造。Cheng和Vigneron [CV07] 提出了另一种基于摩托车图的平板构造方案。为了保持一致性，我们将他们的构造重新表述如下。

Let P denote a simple non-degenerate polygon. Each motorcycle m in M(P) starts from a reflex wavefront vertex v in W (P, 0), which is incident to two wavefront edges. Recall that we call the one wavefront edge that is left to the track of m the left arm of m and the other wavefront edge the right arm of m. To each wavefront edge e, there can exist one motorcycle whose right arm is e and one motorcycle whose left arm is e, see Figure 22.

设P表示一个简单的非退化多边形。M(P)中的每辆摩托车m都从反射波前顶点v开始，该顶点位于W(P, 0)中，并且与两条波前边相连。回想一下，我们将位于m轨迹左侧的波前边称为m的左臂，而另一条波前边称为m的右臂。对于每个波前边e，可能存在一辆摩托车的右臂是e，以及一辆摩托车的左臂是e，参见图22。

Definition 2.13 (lower envelope). Let Cˆ denote the union of e(0) and the tilted traces of the motorcycles that have e as an arm. For each wavefront edge e we define a slab that lies on the supporting plane of fˆ(e) and is bounded from below by Cˆ and two rays at the endpoints of Cˆ that are perpendicular to e(0). The lower envelope of these slabs is denoted by L(P).

定义 2.13 (下包络线)[NT0]。令 Cˆ 表示 e(0) 以及将 e 作为臂的摩托车的倾斜轨迹的并集。对于每个波前边缘 e，我们定义一个位于 fˆ(e) 的支撑平面上的板，该板的下界为 Cˆ，上界为 Cˆ 端点处的两条垂直于 e(0) 的射线。这些板的下包络线表示为 L(P)。

Lemma 2.14 ([CV07]). For a simple non-degenerate polygon P holds L(P) = T (P).

引理 2.14 ([CV07])[NT0]。对于一个简单的非退化多边形P，成立 L(P) = T (P)。
Proof. Consider a point p on T (P). Because p sits on a tilted face fˆ(e) of a wavefront edge, it follows that p is also contained in the corresponding slab of and, thereby, not below L(P). It remains to show that a point p of T (P) is not above L(P). Assume, to the contrary, that p is above any slab of an wavefront edge e, see Figure 22. One can project p down onto the slab and obtain the point u. Then one can project u along the steepest descent of the slab until the lower boundary of the slab is hit at the point v. By Corollary 2.12, the point v is above or just on T (P). Hence, one can project v onto T (P) and obtain q, which is below v or identical to v. The slope of the line pq is at most 1, by Lemma 2.1. On the other hand, the slope of pq is strictly greater than the slope of uv, which is exactly 1. This is a contradiction.

证明。考虑T (P)上的一个点p。因为p位于波前边缘的倾斜面fˆ(e)上，所以p也包含在相应的板条中，因此不低于L(P)。现在需要证明T (P)上的一个点p不高于L(P)。假设相反，p高于波前边缘e的任何板条，见图22。可以将p向下投影到板条上，得到点u。然后可以沿着板条的最速下降方向投影u，直到在点v处到达板条的下边界。根据推论2.12，点v高于或恰好

在T (P)上。因此，可以将v投影到T (P)上，得到q，它低于v或与v相同。根据引理2.1，直线pq的斜率最多为1。另一方面，pq的斜率严格大于uv的斜率，而uv的斜率恰好为1。这是一个矛盾。
Cheng and Vigneron [CV07] pursued the approach of Eppstein and Erickson to define their set of slabs. They reuse the term edge slab from [EE99] and, instead of reflex slabs, they introduce the so-called motorcycle slabs. The approach presented above, however, extends more naturally to arbitrary planar straight-line graphs, see Section 2.4.

Cheng和Vigneron [CV07] 沿用了Eppstein和Erickson的方法来定义他们的板集合。他们沿用了[EE99]中的术语“边板”，并且没有使用反射板，而是引入了所谓的“摩托车板”。然而，上面介绍的方法可以更自然地扩展到任意平面直线图，参见第2.4节。

Let us revisit the discussion concerning an alternative characterization of straight skeletons after Corollary 2.8. Recall that the lower envelope of the slabs of Eppstein and Erickson [EE99] does not lead to an alternative characterization of straight skeletons, because the sizes of the slabs depend on the length of reflex arcs. The slabs of Cheng and Vigneron [CV07], however, do indeed lead to an alternative characterization of straight skeletons: Their slabs only depend on the motorcycle graph. Again, a single slab is not only locally defined by the corresponding wavefront edge. Nevertheless, the contributions of Cheng and Vigneron [CV07] only apply to non-degenerate polygons, whereas the results of Eppstein and Erickson [EE99] apply to arbitrary planar straight-line graphs.

让我们回顾一下推论 2.8 之后关于直线骨架的另一种表征的讨论。回想一下，Eppstein 和 Erickson [EE99] 的板的下包络线并没有引出直线骨架的另一种表征，因为板的大小取决于反射弧的长度。然而，Cheng 和 Vigneron [CV07] 的板确实引出了直线骨架的另一种表征：他们的板仅取决于摩托车图。同样，单个板不仅由相应的波前边缘局部定义。然而，Cheng 和 Vigneron [CV07] 的贡献仅适用于非退化多边形，而 Eppstein 和 Erickson [EE99] 的结果适用于任意平面直线图。

We want to give three reasons why Theorem 2.11 appears to be important. Firstly, the motorcycle graph is an extraction of the “essential sub problem” of computing straight skeletons. Hence, it is important to know the geometric relation between straight skeletons and motorcycle graphs. Secondly, the motorcycle graph helps to devise algorithms to compute the straight skeleton. Thirdly, the motorcycle graph helps to devise an alternative characterization of straight skeletons. Aichholzer and Aurenhammer [AA96] already remarked that it is desirable to find a non-procedural definition of straight skeletons. For this reasons we consider our generalization of Theorem 2.11 to arbitrary planar straight-line graphs in Section 2.4 as important.

我们想给出定理 2.11 显得重要的三个理由。首先，摩托车图是对计算直线骨架的“本质子问题”的提取。因此，了解直线骨架和摩托车图之间的几何关系非常重要。其次，摩托车图有助于设计计算直线骨架的算法。第三，摩托车图有助于设计直线骨架的另一种表征。Aichholzer 和 Aurenhammer [AA96] 已经指出，找到直线骨架的非程序性定义是可取的。由于这些原因，我们将第 2.4 节中定理 2.11 推广到任意平面直线图视为重要。

Cheng and Vigneron [CV07] presented a randomized straight-skeleton algorithm for simple non-degenerate polygons P, which is based on Theorem 2.11, see Section 1.4.2.4. They also present an extension of their algorithm to non-degenerate polygons P with holes. It is easy to see that Theorem 2.11 holds for these slightly more general polygons as well, after generalizing the definition of the motorcycle graph induced by P accordingly.

Cheng和Vigneron [CV07] 提出了一个针对简单非退化多边形 P 的随机化直线骨架算法，该算法基于定理2.11，参见第1.4.2.4节。他们还提出了将其算法扩展到带有孔的非退化多边形 P。很容易看出，在相应地推广由 P 诱导的摩托车图的定义后，定理2.11也适用于这些稍微更一般的多边形。

2.2 t h e t r i a n g u l a t i o n - b a s e d a p p r o a c h

2.2 基于三角剖近的方法

In this section, we study the number of flip events that occur in the triangulation-based straight-skeleton algorithm by Aichholzer and Aurenhammer [AA98]. The upper bound of O(n3) is easy to see, cf. Section 1.4.2.2. However, to the best of our knowledge, no nvertex polygon or planar straight-line graph is known that exceeds a quadratic number of flip events. This circumstance constitutes a gap by a linear factor and the question remains open, whether the number of flip events is actually bound by O(n2). Besides, note that processing edge and split events already takes O(n2 log n) time: Even though there are only Θ(n) edge and split events in total, a single event requires up to O(n log n) time, because the collapsing times of O(n) triangles may need an update in the priority queue. In fact, even a modest convex polygon can lead to a runtime consumption of Θ(n2 log n), as illustrated in Figure 23.

在本节中，我们研究了 Aichholzer 和 Aurenhammer [AA98] 基于三角剖分的直骨架算法中发生的翻转事件的数量。O(n3) 的上限很容易看出，参见第 1.4.2.2 节。然而，据我们所知，目前还没有已知的 n 顶点多边形或平面直线图超过二次数量的翻转事件。这种情况构成了一个线性因子的差距，问题仍然悬而未决，即翻转事件的数量是否实际上受 O(n2) 约束。此外，请注意，处理边和分割事件已经需要 O(n2 log n) 时间：即使总共有只有 Θ(n) 个边和分割事件，单个事件也可能需要高达 O(n log n) 的时间，因为 O(n) 个三角形的塌陷时间可能需要在优先级队列中更新。事实上，即使是一个适度的凸多边形也可能导致 Θ(n2 log n) 的运行时消耗，如图 23 所示。

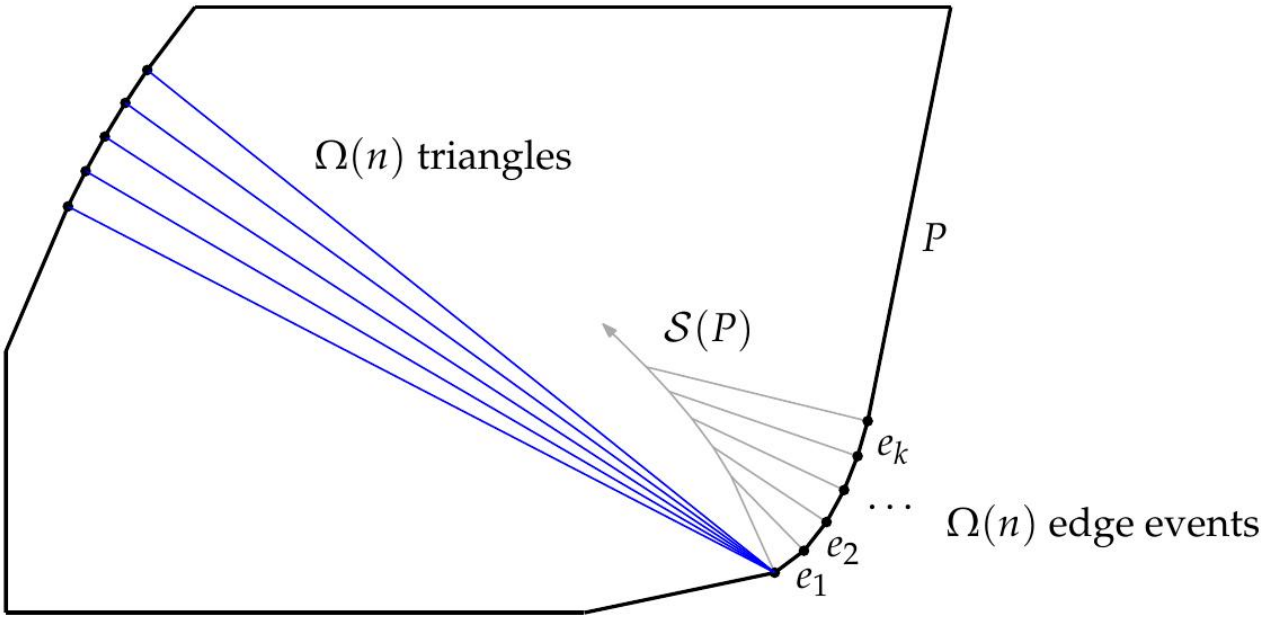


Figure 23: A convex polygon P with n vertices that causes a Θ(n2 log n) runtime for the triangulationbase straight-skeleton algorithm. Triangulation diagonals are shown in blue. Part of the straight skeleton is depicted in gray. The edge events for the edges e1, . . . , ek, with k ∈ Ω(n), occur in the order e1, . . . , ek. For each edge event, the collapsing times of Ω(n) incident triangles have to be updated, which consumes Θ(n2 log n) time in total.

图 23：一个具有 n 个顶点的凸多边形 P，它导致三角剖分基直线骨架算法的运行时间为 Θ(n2 log n)。三角剖分对角线以蓝色显示。部分直线骨架以灰色描绘。边 e1, . . . , ek（其中 k ∈ Ω(n)）的边事件按 e1, . . . , ek 的顺序发生。对于每个边事件，必须更新 Ω(n) 个相邻三角形的塌陷时间，这总共消耗 Θ(n2 log n) 时间。

2.2.1 The number of reappearances of diagonals

2.2.1 对角线重现次数

In Section 1.4.2.2, we explained the O(n3) bound for the number of flip events, by observing that three constantly moving points do not get collinear more than twice and every flip event corresponds to a collinearity of a triple of vertices. But note that not every collinearity does necessarily correspond to a flip event. Let us consider a triangulation diagonal between two vertices A and B. This diagonal may disappear because another vertex S crosses this diagonal during the propagation process. However, it could be possible that the diagonal is restored by subsequent flip events as illustrated in Figure 24. In order to have the diagonal AB restored, the vertex S has to back off such that A and B see each other again, as illustrated in Step 2. Hence, the vertices A, B and S got collinear twice and S cannot cause a flip event for AB again. In other words, S cannot flip the diagonal AB twice. Nevertheless, Figure 24 does not prove that the diagonal AB can be actually restored. We just considered the necessary topological changes and not the proper geometric setting that leads to these topological changes. How often can a single diagonal, say AB, actually reappear? If this number would be in O(1) then we would conclude that the number of flip events is in fact in O(n2), because there are only (n2)

pairs of vertices that can be connected by a diagonal. Unfortunately, the following lemma does not even state that a single diagonal can reappear Ω(n) times, but Ω(n) diagonals can each reappear Ω(n) times, which leads to Ω(n2) flip events in total.

在第1.4.2.2节中，我们解释了翻转事件数量的O(n3)界限，通过观察到三个恒定移动的点不会共线超过两次，并且每个翻转事件都对应于一个顶点三元组的共线性。但请注意，并非每个共线性都必然对应于一个翻转事件。让我们考虑两个顶点A和B之间的三角剖分对角线。这条对角线可能会因为另一个顶点S在传播过程中穿过这条对角线而消失。然而，如图24所示，这条对角线可能会被后续的翻转事件恢复。为了使对角线AB恢复，顶点S必须后退，使得A和B再次相互看到，如步骤2所示。因此，顶点A、B和S共线了两次，并且S不能再次引起AB的翻转事件。换句话说，S不能翻转对角线AB两次。然而，图24并不能证明对角线AB实际上可以被恢复。我们仅仅考虑了必要的拓扑变化，而没有考虑导致这些拓扑变化的适当几何设置。一条单独的对角线，比如AB，实际上可以重新出现多少次？如果这个数字是O(1)，那么我们可以得出结论，翻转事件的数量实际上是O(n2)，因为只有(n2)对顶点可以通过对角线连接。不幸的是，下面的引理甚至没有说明一条单独的对角线可以重新出现Ω(n)次，而是Ω(n)条对角线可以各自重新出现Ω(n)次，这导致总共有Ω(n2)个翻转事件。

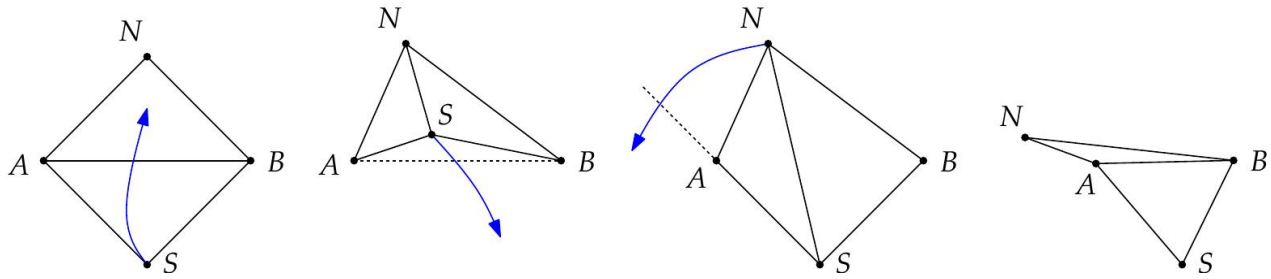


Figure 24: A sequence of flip events which leads to the reappearance of the diagonal AB between the vertices A and B.

图 24：导致顶点 A 和 B 之间对角线 AB 重新出现的一系列翻转事件。

Lemma 2.15 ([HH10b]). There exist polygons P with n vertices and triangulations T of P such that Ω(n) diagonals each reappear Ω(n) times during the wavefront propagation.

引理 2.15 ([HH10b]).存在具有 n 个顶点的多边形 P 以及 P 的三角剖分 T，使得在波前传播期间，每个 Ω(n) 对角线重新出现 Ω(n) 次。

Proof. We prove the statement in three steps. In each step we give a geometric setting of moving vertices, for which a specific sequence of flip events occurs. At first, we describe how a single diagonal AB reappears twice. In the second step, we extend the construction such that AB reappears Ω(n) times. In the third step, we show how k diagonals A1B, . . . , AkB reappear each Ω(n) times, with k ∈ Ω(n).

证明。我们分三步证明该陈述。在每个步骤中，我们都给出一个移动顶点的几何设置，其中发生一个特定的翻转事件序列。首先，我们描述单个对角线AB如何重新出现两次。在第二步中，我们扩展该构造，使得AB重新出现Ω(n)次。在第三步中，我们展示了如何使k条对角线A1B, . . . , AkB各自重新出现Ω(n)次，其中k ∈ Ω(n)。

Let us consider six vertices A, B, S0, S1, N0, N1. The goal is to carefully construct start points and velocities for these six vertices such that the topological transitions, as illustrated in Figure 25, are executed. We denote by V(t) the position of a vertex V at time t. The basic building block of this proof works as follows: We want that the vertex S0 flips the diagonal AB and after that backs off such that the vertices A and B see each other again. All vertices are bound to move with constant speed. We let A and B drive vertically upwards and we assume that A(0) is at the origin and B(0) at (1, 0). We denote by vA and vB the speeds of A and B and we further assume that vA = 2 and $1 < vB/vA \ll 2$.

我们考虑六个顶点A, B, S0, S1, N0, N1。目标是仔细构造这六个顶点的起始点和速度，以便执行如图25所示的拓扑转换。我们用V(t)表示顶点V在时间t的位置。这个证明的基本组成部分如下：我们希望顶点S0翻转对角线AB，然后退回，使得顶点A和B再次彼此可见。所有顶点都以恒定速度移动。我们让A和B垂直向上移动，并假设A(0)位于原点，B(0)位于(1, 0)。我们用vA和vB表示A和B的速度，并进一步假设vA = 2和$1 < vB/vA \ll 2$。

The movement of the vertex S0 is completely determined by choosing the loci of S0(1) and S0(1 + Δ), where $\Delta \in (0, 1/4)$ is fixed. The locus of S0(1) is chosen to be on the straightline segment [A(1)B(1)] and S0(1 + Δ) is chosen to be on [A(1 + Δ)B(1 + Δ)]. We further require that S0 moves to the north-east, i. e. that S0(1 + Δ) lies strictly to the right to S0(1). The determinant det(A(t), B(t), S0(t)), which is

顶点S0的运动完全由选择S0(1)和S0(1 + Δ)的轨迹决定，其中$\Delta \in (0, 1/4)$是固定的。S0(1)的轨迹被选择在直线段[A(1)B(1)]上，而S0(1 + Δ)被选择在[A(1 + Δ)B(1 + Δ)]上。我们进一步要求S0向东北方向移动，即S0(1 + Δ)严格位于S0(1)的右侧。行列式det(A(t), B(t), S0(t))，即

A

x

(

t
)
B
x
(
t
)
S
0,
x
(
t
)
A
y
(
t
)
B
y
(
t
)
S

0,
y
(
t
)
1
1
1
□
□
□
□
□
□
,
(2.3)
is a quadratic polynomial in t and its sign corresponds to the orientation of the three vertices. (The x- and y-coordinates of the vertices A, B, S0 are denoted by subscripts.) We observe that for some $t > 1 + \Delta$ the points A(t), B(t), S0(t) are in clockwise position. This is because the vertex S0 moves to north-east and vB/vA > 1. Knowing that det(A(t), B(t), S0(t)) is a quadratic polynomial with roots $\{1, 1 + \Delta\}$, and by observing that A(t), B(t), S0(t) are in clockwise position for some $t > 1 + \Delta$, we conclude that the triangle A(t), B(t), S0(t) is clockwise for all t < 1, counter-clockwise for all $t \in (1, 1 + \Delta)$ and clockwise again for all $t > 1 + \Delta$. In other words, we constructed a vertex S0 which indeed executes the first two transitions in Figure 25.

是t的二次多项式，其符号对应于三个顶点的方向。（顶点A, B, S0的x和y坐标用下标表示。）我们观察到，对于某些$t > 1 + \Delta$，点A(t), B(t), S0(t)呈顺时针位置。这是因为顶点S0向东北方向移动，且vB/vA > 1。已知det(A(t), B(t), S0(t))是一个以$\{1, 1 + \Delta\}$为根的二次多项式，并且通过观察得知对于某些$t > 1 + \Delta$，A(t), B(t), S0(t)呈顺时针位置，我们得出结

论：三角形A(t), B(t), S0(t)对于所有t < 1呈顺时针方向，对于所有t ∈ (1, 1 + ∆)呈逆时针方向，对于所有t > 1 + ∆再次呈顺时针方向。换句话说，我们构造了一个顶点S0，它确实执行了图25中的前两个转换。

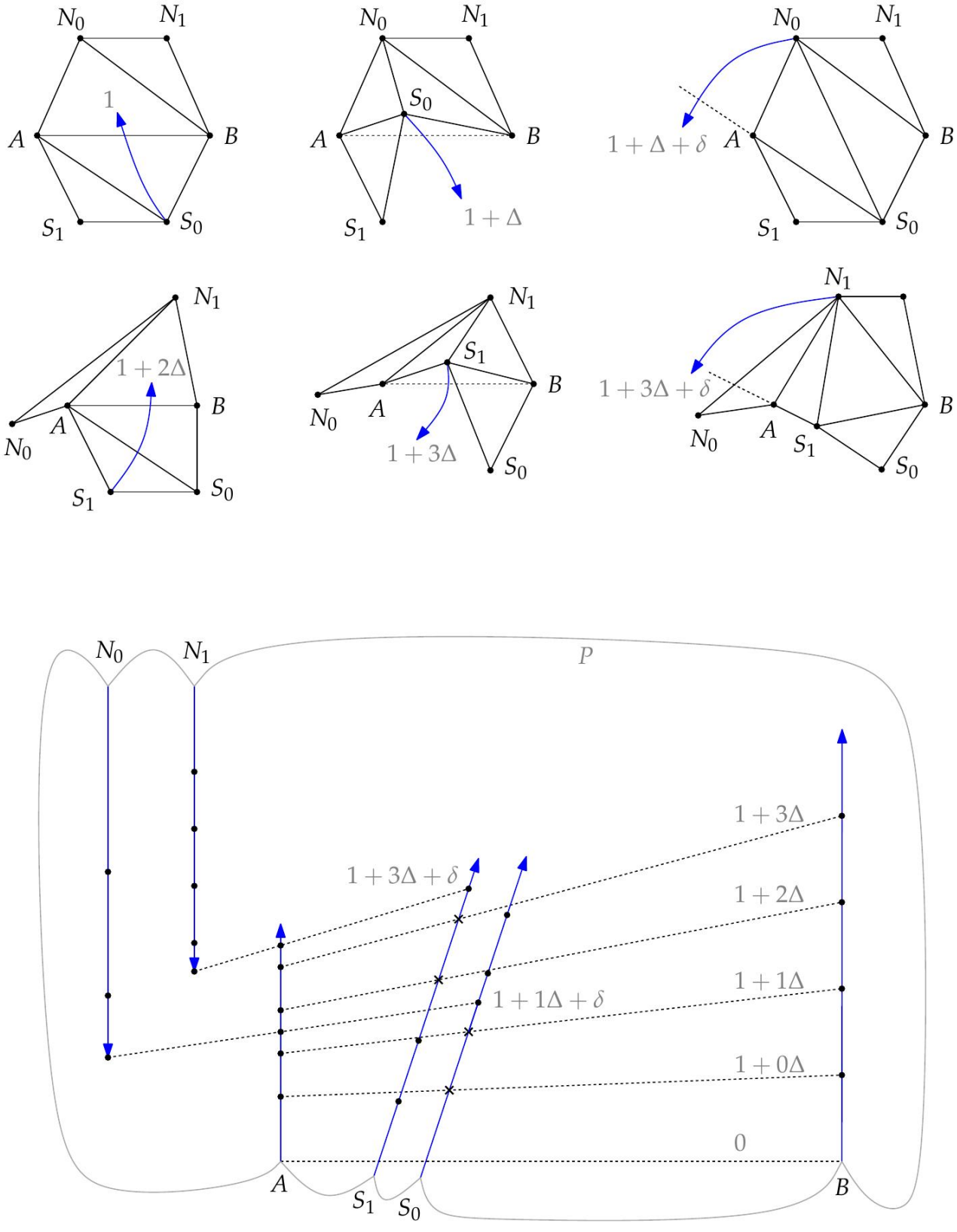


Figure 25: Step 1 of the proof of Lemma 2.15: The diagonal AB reappears twice since the geometric setting at the bottom produces the sequence of flip events at the top. Bottom: Every black dot denotes the position of a vertex at the time given which is depicted in

gray. A cross mark indicates collinearity for A and B with S0 resp. S1. Top: Each figure illustrates a topological transition. The second reappearance is caused by the sixth transition.

图 25：引理 2.15 证明的步骤 1：对角线 AB 重新出现两次，因为底部的几何设置产生了顶部的翻转事件序列。底部：每个黑点表示给定时间的顶点位置，以灰色描绘。交叉标记表示 A 和 B 分别与 S0 和 S1 共线。顶部：每个图示都说明了一个拓扑转变。第二次重新出现是由第六次转变引起的。

According to our desired sequence of transitions, we need a vertex N0 such that N0(t) gets collinear with A(t) and S0(t) for some $t > 1 + \Delta$, say at $t = 1 + \Delta + \delta$, where $\delta \in (0, \Delta/2)$ is fixed. (In the sequel, we have to choose δ small enough.) We place a vertex N0 that moves southwards and parallel to A. Furthermore, we choose N0,y(0) = 8 such that N0(0) is strictly above B(1 + 4Δ). Next, we request that N0(1 + Δ + δ) is on the supporting line A(1 + Δ + δ) B(1 + Δ + δ). Hence, at time 1 + Δ + δ the vertex N0 causes the recreation of the diagonal AB by flipping the diagonal N0S0.

根据我们期望的转换序列，我们需要一个顶点N0，使得N0(t)与A(t)和S0(t)共线，对于某个$t > 1 + \Delta$，比如在$t = 1 + \Delta + \delta$时，其中$\delta \in (0, \Delta/2)$是固定的。（在下文中，我们必须选择足够小的δ。）我们放置一个顶点N0，它向南移动且平行于A。此外，我们选择N0,y(0) = 8，使得N0(0)严格位于B(1 + 4Δ)之上。接下来，我们要求N0(1 + Δ + δ)位于支撑线A(1 + Δ + δ) B(1 + Δ + δ)上。因此，在时间1 + Δ + δ时，顶点N0通过翻转对角线N0S0来导致对角线AB的重建。

In order to conclude the first step of the proof, we repeat the life-cycle of the diagonal AB, by using the vertices S1, N1 instead of S0, N0. We place the vertex S1 between A and S0 and let S1 move parallel to S0. The locus of S1(t) is chosen to be on [A(t)B(t)] for $t \in \{1 + 2\Delta, 1 + 3\Delta\}$. Applying the same argument as for S0, we observe that the triangle A(t), B(t), S1(t) is clockwise for all $t < 1 + 2\Delta$, counter-clockwise for all $t \in (1 + 2\Delta, 1 + 3\Delta)$ and clockwise again for all $t > 1 + 3\Delta$. Note that S0 is already behind A(t)B(t) for $t > 1 + \Delta$. Hence, S1 causes a flip event for the diagonal AB at $t = 1 + 2\Delta$ and falls back at $t = 1 + 3\Delta$ such that the vertices A and B see each other again for $t > 1 + 3\Delta$. Also note that the introduction of S1 does not interfere with the transitions caused by S0 and N0, since S1 is behind the supporting line of A, B, S0 at $t = 1 + \Delta$. Hence, by choosing δ small enough, S1 is also behind the supporting line of A and S0 at $t = 1 + \Delta + \delta$.

为了结束证明的第一步，我们重复对角线AB的生命周期，通过使用顶点S1, N1来代替S0, N0。我们将顶点S1放置在A和S0之间，并让S1平行于S0移动。S1(t)的轨迹被选择在[A(t)B(t)]上，对于$t \in \{1 + 2\Delta, 1 + 3\Delta\}$。应用与S0相同的论证，我们观察到三角形A(t), B(t), S1(t)对于所有$t < 1 + 2\Delta$是顺时针方向，对于所有$t \in (1 + 2\Delta, 1 + 3\Delta)$是逆时针方向，并且对于所有$t > 1 + 3\Delta$再次是顺时针方向。请注意，对于$t > 1 + \Delta$，S0已经位于A(t)B(t)之后。因此，S1在$t = 1 + 2\Delta$时引起对角线AB的翻转事件，并在$t = 1 + 3\Delta$时回落，使得顶点A和B对于$t > 1 + 3\Delta$再次彼此可见。另请注意，S1的引入不会干扰由S0和N0引起的转换，因为在$t = 1 + \Delta$时，S1位于A, B, S0的支持线之后。因此，通过选择足够小的δ，在$t = 1 + \Delta + \delta$时，S1也位于A和S0的支持线之后。

Finally, we place a vertex N1 between N0 and A which again moves southwards and parallel to A. We assume that N0 and N1 start from the same horizontal line and we require that N1(1 + 3∆ + δ) is collinear with A(1 + 3∆ + δ) and S1(1 + 3∆ + δ). Hence, the diagonal AB is recreated again by an edge flip of the diagonal N1S1.

最后，我们在N0和A之间放置一个顶点N1，它再次向南移动并平行于A。我们假设N0和N1从同一水平线开始，并且我们要求N1(1 + 3∆ + δ)与A(1 + 3∆ + δ)和S1(1 + 3∆ + δ)共线。因此，对角线AB通过对角线N1S1的边翻转再次被重新创建。

To sum up, we are able to construct a geometric setting of moving vertices such that the diagonal between the vertices A and B reappears twice. However, in order to finish Step 1 of the proof, we have to guarantee that there is a polygon P and a triangulation T of P that realizes the sequence of transitions in Figure 25. First, we note that we can choose ∆ very small such that the speeds of N0 and N1 get arbitrarily close. Next, we note that if we choose vB/vA > 1 but arbitrarily close to 1 then the points S1(0) and S2(0) approach the supporting line of A(0) and B(0). Hence, we can construct a polygon P, as in Figure 25, such that only the vertices A, S1, S0, B, N1, N0 of P are reflex and connected by convex chains. By choosing the incident polygon edges of each of these vertices accordingly, we obtain the desired velocities for the propagating wavefront vertices. The initial triangulation T of P contains the edges given in Figure 25 and the remaining faces can be triangulated arbitrarily.

总而言之，我们能够构建一个移动顶点的几何设置，使得顶点A和B之间的对角线重新出现两次。然而，为了完成证明的步骤1，我们必须保证存在一个多边形P和一个P的三角剖分T，其实现了图25中的变换序列。首先，我们注意到我们可以选择非常小的∆，使得N0和N1的速度任意接近。接下来，我们注意到，如果我们选择vB/vA > 1但任意接近1，那么点S1(0)和S2(0)接近A(0)和B(0)的支撑线。因此，我们可以构造一个多边形P，如图25所示，使得P中只有顶点A, S1, S0, B, N1, N0是凹的，并且由凸链连接。通过相应地选择每个这些顶点的入射多边形边，我们获得了传播波前顶点的期望速度。P的初始三角剖分T包含图25中给出的边，剩余的面可以任意三角剖分。

For Step 2 we extend our construction by adding vertices S2, . . . , Sm from right to left between S1 and A, with m ∈ Ω(n). We repeat the construction scheme presented above for these new vertices. Furthermore, we choose ∆ < 1/2m+2 in order to make the new vertices fit. Likewise we add vertices N2, . . . , Nm from left to right between N1 and A. In general, for each i ∈ {0, . . . , m} the vertex Si causes a flip event for the diagonal AB at time 1 + 2i∆ and falls back behind A(t) B(t) at time t = 1 + (2i + 1)∆. At time t = 1 + (2i + 1)∆ + δ the vertex Ni gets collinear with A(t) and Si(t) and causes the recreation of the diagonal AB.

对于步骤2，我们通过从右到左在S1和A之间添加顶点S2, . . . , Sm来扩展我们的构造，其中m ∈ Ω(n)。我们对这些新顶点重复上述构造方案。此外，我们选择∆ < 1/2m+2，以使新顶点能够适应。同样，我们在N1和A之间从左到右添加顶点N2, . . . , Nm。一般来说，对于每个i ∈ {0, . . . , m}，顶点Si在时间1 + 2i∆时导致对角线AB发生翻转事件，并在时间t = 1 + (2i + 1)∆时落回A(t) B(t)之后。在时间t = 1 + (2i + 1)∆ + δ时，顶点Ni与A(t)和Si(t)共线，并导致对角线AB的重建。

The basic idea of Step 3 is to rename the vertex A to A1 and to carefully place copies A2, . . . , Ak of A1 from right to left. In the following, we only consider a single reappearance cycle for each diagonal A1B, . . . , AkB, caused by flip events due to vertex Si and the subsequent restoration due to vertex Ni, for an arbitrary i ∈ {0, . . . , m}. We illustrated the topo-

步骤 3 的基本思想是将顶点 A 重命名为 A1，并小心地将 A1 的副本 A2, . . . , Ak 从右向左放置。在下文中，我们仅考虑每个对角线 A1B, . . . , AkB 的单个重现周期，该周期由顶点 Si 引起的翻转事件以及随后由顶点 Ni 引起的恢复所导致，其中 i ∈ {0, . . . , m} 是任意的。我们说明了拓扑结构

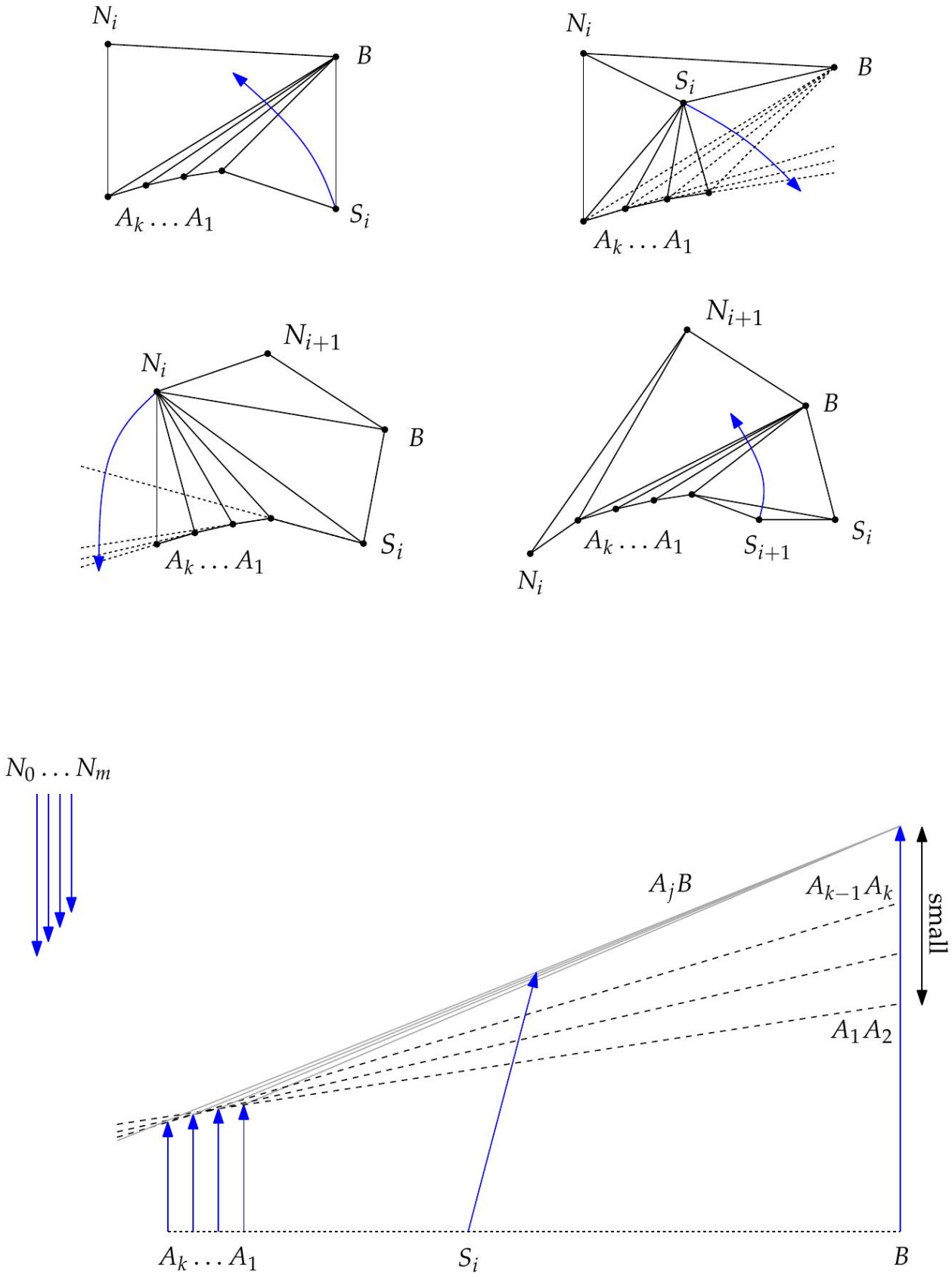


Figure 26: Step 3 of the proof of Lemma 2.15: We place copies of A1 in order to obtain diagonals A1B, . . . , AkB. By arranging the vertices A2, . . . , Ak almost collinear with A1B the diagonals A2B, . . . , AkB behave like the diagonal A1B from Step 2.

图 26：引理 2.15 的证明的第 3 步：我们放置 A1 的副本，以便获得对角线 A1B, . . . , AkB。通过将顶点 A2, . . . , Ak 排列成几乎与 A1B 共线，对角线 A2B, . . . , AkB 的行为类似于第 2 步中的对角线 A1B。
logical transitions in Figure 26. Let us assume that the points B(0), A1(0), . . . , Ak(0) are arranged on a horizontal line, see Figure 26. Furthermore, assume that B(1), A1(1), . . . , Ak(1) are almost on a straight line, by arranging A2(1), . . . , Ak(1) accordingly. However, for any $j \in \{1, \ldots, k-1\}$ each Aj+1(1) shall properly see the vertex B(1) and every supporting line Aj(1)Aj+1(1) shall intersect the trajectory of B in an ascending order. Moreover, the intersection points shall be below but arbitrarily close to B(1), see Figure 26. The idea is to arrange A2(1), . . . , Ak(1) very close to A1(1) B(1) such that the diagonals A1B, . . . , AkB behave almost the same as the single diagonal AB from Step 2. (Note that if B, A1, . . . , Ak are collinear at time 1 then they are collinear at all times, due to the intercept theorems.)

图 26 中的逻辑转换。假设点 B(0), A1(0), . . . , Ak(0) 排列在一条水平线上，参见图 26。此外，假设通过相应地排列 A2(1), . . . , Ak(1)，使得 B(1), A1(1), . . . , Ak(1) 几乎在一条直线上。然而，对于任何 $j \in \{1, \ldots, k-1\}$，每个 Aj+1(1) 应该能适当地看到顶点 B(1)，并且每条支撑线 Aj(1)Aj+1(1) 应该以递增的顺序与 B 的轨迹相交。此外，交点应低于 B(1) 但任意接近 B(1)，参见图 26。其思想是使 A2(1), . . . , Ak(1) 非常接近 A1(1) B(1)，使得对角线 A1B, . . . , AkB 的行为几乎与步骤 2 中的单条对角线 AB 相同。（请注意，如果 B, A1, . . . , Ak 在时间 1 时共线，那么由于截距定理，它们在所有时间都共线。）

We recall from Step 2 that det(A1(t), B(t), Si(t)) is a quadratic polynomial in t with roots $\{1 + 2i\Delta, 1 + (2i + 1)\Delta\}$. Furthermore, A1(t), B(t), Si(t) is in clockwise position for all $t < 1 + 2i\Delta$, in counter-clockwise position for $t \in (1 + 2i\Delta, 1 + (2i + 1)\Delta)$ and again in clockwise position for $t > 1 + (2i + 1)\Delta$. Consider $\tau \in (0, \delta)$ to be fixed. It follows that the triangle A1(t)B(t)Si(t) is in a strict counter-clockwise position for $t = 1 + 2i\Delta + \tau$. Recall that Si crosses the diagonal A1B at $t = 1 + 2i\Delta$. We now choose B(1), A1(1), . . . , Ak(1) as close to collinear as possible such that Si also crosses the diagonal AjB before $t = 1 + 2i\Delta + \tau$ for all $j \in \{1, \ldots, k\}$. That is, Si causes a flip event for each diagonal A1B, . . . , AkB within a temporal tolerance of τ after $t = 1 + 2i\Delta$. Hence, the first topological transition in Figure 26 happened after $t = 1 + 2i\Delta + \tau$.

我们从步骤 2 回忆起，det(A1(t), B(t), Si(t)) 是关于 t 的二次多项式，其根为 $\{1 + 2i\Delta, 1 + (2i + 1)\Delta\}$。此外，对于所有 $t < 1 + 2i\Delta$，A1(t), B(t), Si(t) 处于顺时针位置；对于 $t \in (1 + 2i\Delta, 1 + (2i + 1)\Delta)$，处于逆时针位置；对于 $t > 1 + (2i + 1)\Delta$，再次处于顺时针位置。考虑固定 $\tau \in (0, \delta)$。由此可知，对于 $t = 1 + 2i\Delta + \tau$，三角形 A1(t)B(t)Si(t) 处于严格的逆时针位置。回想一下，Si 在 $t = 1 + 2i\Delta$ 时穿过对角线 A1B。现在我们选择 B(1), A1(1), . . . , Ak(1)，使其尽可能接近共线，使得对于所有 $j \in \{1, \ldots, k\}$，Si 也在 $t = 1 + 2i\Delta + \tau$ 之前穿过对角线 AjB。也就是说，在 $t = 1 + 2i\Delta$ 之后，Si 在 τ 的时间容差内，为每个对角线 A1B, . . . , AkB 引起一次翻转事件。因此，图 26 中的第一次拓扑转变发生在 $t = 1 + 2i\Delta + \tau$ 之后。

According to the second transition in Figure 26, we need Si to fall back behind the former diagonals A1B, . . . , AkB. However, in addition we require that Si also falls behind the supporting lines of Aj Aj+1 for all $j \in \{1, \ldots, k-1\}$. This circumstance leads to flip events

of the diagonals A2Si, . . . , Ak−1Si such that all A1, . . . Ak are connected to Ni. From Step 2 we know that Si falls behind A1B at time t = 1 + (2i + 1)∆. For Step 3 we choose B(1), A1(1), . . . , Ak(1) as close to collinear as necessary such that Si falls behind the supporting lines Aj Aj+1 until t = 1 + (2i + 1)∆ + τ.

根据图 26 中的第二个过渡，我们需要 Si 落后于之前的对角线 A1B, . . . , AkB。然而，此外我们还要求 [NT3]Si[/NT1] 也落后于所有 j ∈ {1, . . . , k − 1} 的 Aj Aj+1 的支撑线。这种情况导致对角线 A2Si, . . . , Ak−1Si 发生翻转事件，使得所有 A1, . . . Ak 都连接到 Ni。从步骤 2 我们知道，在时间 t = 1 + (2i + 1)∆ 时，Si 落后于 A1B。对于步骤 3，我们选择 B(1), A1(1), . . . , Ak(1) 尽可能接近共线，使得 Si 落后于支撑线 Aj Aj+1，直到 t = 1 + (2i + 1)∆ + τ。

In order to execute the third transition in Figure 26, we finally require that Ni crosses the supporting lines Aj Aj+1 until t = 1 + (2i + 1)∆ + δ + τ. After that, all diagonals A1B, . . . , AkB are restored and we completed the reappearance cycle caused by Si and Ni.

为了执行图26中的第三个转换，我们最终要求Ni穿过支撑线Aj Aj+1，直到t = 1 + (2i + 1)∆ + δ + τ。之后，所有对角线A1B, . . . , AkB都被恢复，并且我们完成了由Si和Ni引起的重现周期。

To sum up, we can arrange A2(1), . . . , Ak(1) as close as necessary to A1(1) B(1) such that the topological transitions in Figure 26 are executed for all Si, Ni, with 0 ≤ i ≤ m, within a temporal tolerance of τ ∈ (0, δ). Furthermore, we observe that at time zero the vertices B, A1, . . . , Ak are on a horizontal line. In order to enable diagonals A1B, . . . , AkB in the initial triangulation, we start our simulation at time −ϵ, for a sufficiently small ϵ > 0.

总而言之，我们可以将A2(1), . . . , Ak(1)排列得尽可能接近A1(1) B(1)，使得图26中的拓扑转换对于所有Si, Ni，其中0 ≤ i ≤ m，在τ ∈ (0, δ)的时间容差内执行。此外，我们观察到在零时刻，顶点B, A1, . . . , Ak位于一条水平线上。为了在初始三角剖分中启用对角线A1B, . . . , AkB，我们在时间−ε开始我们的模拟，其中ε > 0足够小。

2.2.2 Good triangulations and bad polygons

2.2.2 好的三角剖分和坏的多边形

As a byproduct of Lemma 2.15, we obtain that polygons P with n vertices and corresponding triangulations exist for which Ω(n2) flip events occur. Besides the actual shape of the polygon, this result also hinges on the specific initial triangulation. If the initial triangulation from Lemma 2.15 would not contain the diagonals A1B, . . . AkB, but, for example, the diagonals S0Nm, . . . , Sm Nm then we would easily avoid the occurrence of Ω(n2) flip events. Can we always find for a polygon P a “good” initial triangulation, for which only a few number of flip events occur — say, at most o(n2) or even O(n)? Or, conversely, are there “bad” polygons, for which any triangulation ends up with a large number of flip events?

作为引理 2.15 的副产品，我们得到存在具有 n 个顶点的多边形 P 以及相应的三角剖分，对于这些三角剖分，会发生 Ω(n2) 次翻转事件。除了多边形的实际形状外，这个结果还取决于特定的初始三角剖分。如果引理 2.15 中的初始三角剖分不包含对角线 A1B, . . . AkB，

而是包含例如对角线 S0Nm, . . . , Sm Nm，那么我们可以很容易地避免发生 Ω(n2) 次翻转事件。我们是否总能为一个多边形 P 找到一个“好的”初始三角剖分，使得只发生少数几次翻转事件——比如，最多 o(n2) 甚至 O(n) 次？或者，相反，是否存在“坏的”多边形，对于这些多边形，任何三角剖分最终都会导致大量的翻转事件？

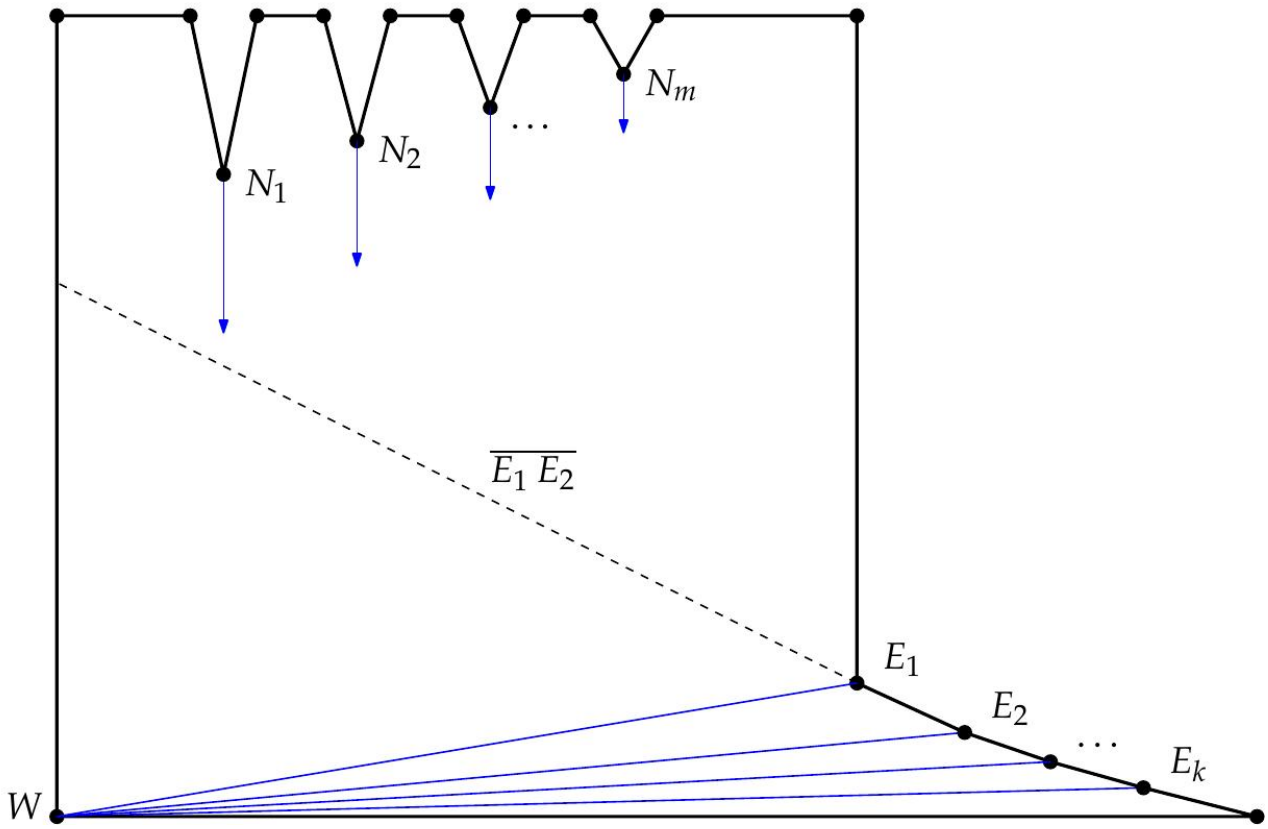


Figure 27: An n-vertex polygon for which any triangulation leads to Ω(n2) flip events.

图 27：一个 n 顶点多边形，对于该多边形，任何三角剖分都会导致 Ω(n2) 次翻转事件。

Lemma 2.16 ([HH10b]). There exist polygons P with n vertices, for which any triangulation leads to Ω(n2) flip events.

引理 2.16 ([HH10b]).存在具有n个顶点的多边形P，对于这些多边形，任何三角剖分都会导致Ω(n2)次翻转事件。

Proof. Let us consider the polygon illustrated in Figure 27. The polygon has k ∈ Ω(n) vertices E1, . . . , Ek that form a reflex chain. The supporting line E1 E2 is chosen in a way such that E2, . . . , Ek only see the vertex W (except for their neighboring vertices, of course). Hence, any triangulation for P necessarily contains the diagonals E1W, . . . , EkW. The idea is that each of these Ω(n) diagonals is flipped Ω(n) times by vertices N1, . . . , Nm, with m ∈ Ω(n). We choose the vertex Ni fast enough such that it leads to a split event with the bottom polygon edge before Ni+1 crosses E1 E2. Furthermore, we make the vertices N1, . . . , Nm fast enough such that the split events for N1, . . . , Nm happen before any other split or edge event happens.

证明。 让我们考虑图27中所示的多边形。该多边形有k ∈ Ω(n)个顶点E1, . . . , Ek，它们形成一个凹链。支撑线E1 E2的选择方式使得E2, . . . , Ek仅看到顶点W（当然，除了它们的相邻顶点）。因此，P的任何三角剖分必然包含对角线E1W, . . . , EkW。这个想法是，这些Ω(n)条对角线中的每一条都被顶点N1, . . . , Nm翻转Ω(n)次，其中m ∈ Ω(n)。我们选择顶点Ni的速度足够快，以便在Ni+1穿过E1 E2之前，它导致与底部多边形边的分裂事件。此外，我们使顶点N1, . . . , Nm的速度足够快，以便N1, . . . , Nm的分裂事件发生在任何其他分裂或边事件发生之前。
If we choose the speeds of N1, . . . , Nm accordingly then we observe that N1 leads to Ω(n) flip events with E1W, . . . , EkW and we obtain diagonals N1E1, . . . N1, Ek. Next, the vertex N2 flips the diagonals N1E1, . . . , N1Ek and we obtain diagonals N2E1, . . . , N2Ek. (In the meanwhile N1 could have caused a split event already. However, the corresponding diagonals remain incident to a wavefront vertex emanated from this split event and will be flipped by N2.) At the end, each vertex N1, . . . , Nm will flip Ω(n) diagonals and we obtain Ω(n2) flip events in total.

如果我们相应地选择 N1, . . . , Nm 的速度，那么我们观察到 N1 导致 Ω(n) 次翻转事件，其中包含 E1W, . . . , EkW，并且我们获得对角线 N1E1, . . . N1, Ek。接下来，顶点 N2 翻转对角线 N1E1, . . . , N1Ek，并且我们获得对角线 N2E1, . . . , N2Ek。（与此同时，N1 可能已经导致了一个分裂事件。然而，相应的对角线仍然与从该分裂事件发出的波前顶点相关联，并将被 N2.) 翻转。）最后，每个顶点 N1, . . . , Nm 将翻转 Ω(n) 条对角线，并且我们总共获得 Ω(n2) 次翻转事件。
In order to reduce the number of flip events it seems reasonable to try to re-triangulate the wavefront W (P, t) at favorable moments. In other words, we could try to pay the costs for computing a new triangulating in exchange to the costs caused for a certain number of flip events. However, we want to remark that one re-triangulation of the polygon constructed in the proof of Lemma 2.16 at any point in time saves at most O(n) flip events: If Ni is already below E1 E2 then all vertices N1, . . . , Ni−1 already caused Ω((i − 1)n) flip events and Ni+1 is still above E1 E2. Hence, we avoid at most O(n) flip events that would be caused by Ni itself and pay the costs for a re-triangulation.

为了减少翻转事件的数量，在有利的时刻尝试重新三角化波前 W (P, t) 似乎是合理的。换句话说，我们可以尝试支付计算新三角剖分的成本，以换取一定数量的翻转事件所造成的成本。然而，我们想指出的是，在引理 2.16 的证明中，在任何时间点重新三角化多边形最多可以节省 O(n) 个翻转事件：如果 Ni 已经低于 E1 E2，那么所有顶点 N1, . . . , Ni−1 已经导致 Ω((i − 1)n) 个翻转事件，并且 Ni+1 仍然高于 E1 E2。因此，我们最多避免 Ni 本身造成的 O(n) 个翻转事件，并支付重新三角剖分的成本。

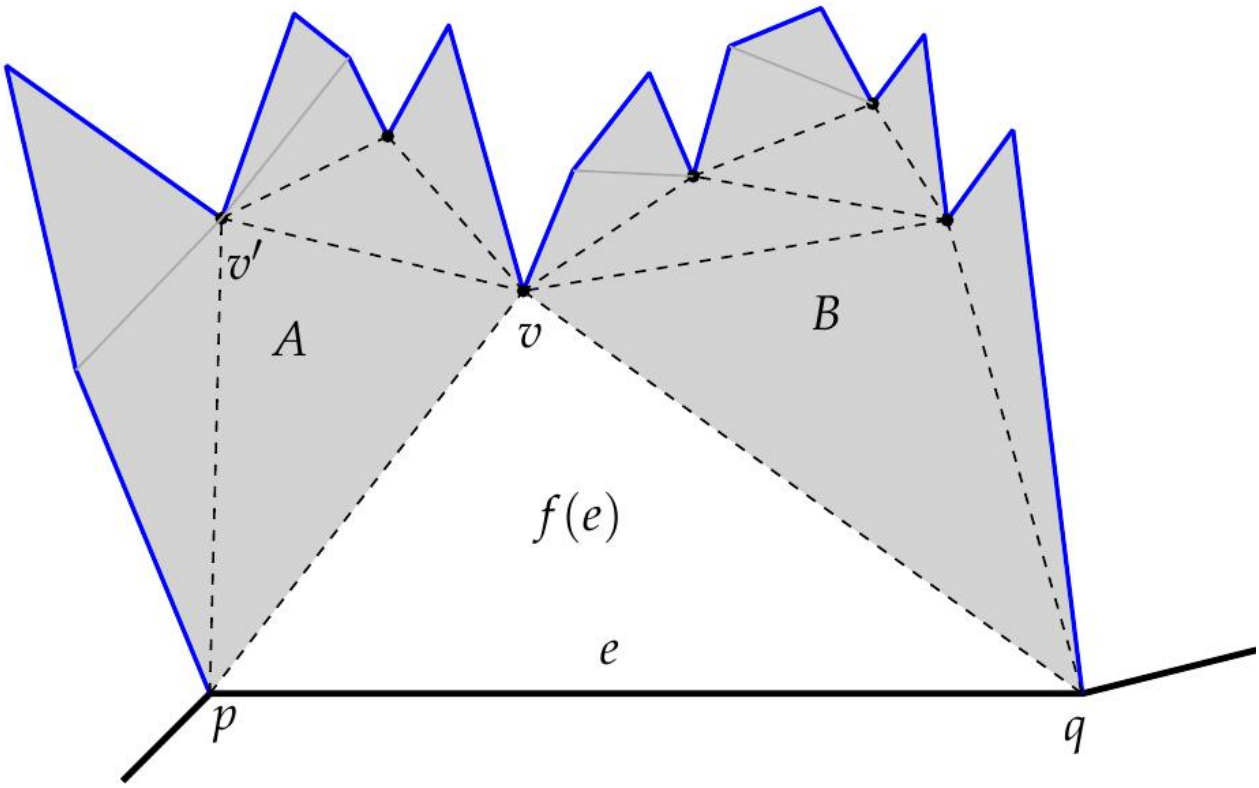


Figure 28: A recursive triangulation scheme for the face f (e) which is free of flip events.

图 28：面 f (e) 的递归三角剖分方案，该方案没有翻转事件。

2.2.3 Steiner triangulations without flip events

2.2.3 无翻转事件的斯坦纳三角剖分

Lemma 2.16 tells us that ordinary triangulations do not allow a strategy to keep the number of flip events below O(n2) for arbitrary polygons. However, we observe that if we use Steiner[1] points, we could reduce the number of flip events significantly for the polygon illustrated in Figure 27. Nevertheless, introducing Steiner points to the initial triangulation in the algorithm by Aichholzer and Aurenhammer [AA98] obtrudes several questions: (i) how do Steiner points interact with the wavefront, (ii) how do we determine the loci of the Steiner points and (iii) how effective is the introduction of Steiner points in order to reduce the number of flip events? Regarding the first question, we assume that the Steiner points keep still. Furthermore, it seems natural to maintain the property that the triangulation keeps the area R2 \ St′≤t W (P, t′) triangulated for any time t ≥ 0. We give an answer to the remaining two questions by the following lemma.

引理 2.16 告诉我们，对于任意多边形，普通三角剖分不允许将翻转事件的数量保持在 O(n2) 以下的策略。然而，我们观察到，如果我们使用斯坦纳[1]点，我们可以显著减少图 27 所示多边形的翻转事件数量。然而，在 Aichholzer 和 Aurenhammer [AA98] 的算法中，将斯坦纳点引入初始三角剖分会产生几个问题：（i）斯坦纳点如何与波前相互作用，（ii）我们如何确定斯坦纳点的位置，以及（iii）引入斯坦纳点在减少翻转事件数量方面的效果如何？关于第一个问题，我们假设斯坦纳点保持静止。此外，保持三角剖分保持区域 R2 \ St′≤t W (P, t′) 在任何时间 t ≥ 0 内三角化的性质似乎是很自然的。我们通过以下引理给出剩余两个问题的答案。

Lemma 2.17 ([HH10b]). Every simple polygon P with n vertices admits a triangulation that employs at most n − 2 Steiner points and is free of flip events.

引理 2.17 ([HH10b]).每个具有 n 个顶点的简单多边形 P 允许进行三角剖分，该剖分最多使用 n − 2 个斯坦纳点，并且没有翻转事件。

Proof. We prove the statement by first presenting a proper set of Steiner points and a proper triangulation. In the sequel we show that no flip events occur for that particular triangulation.

证明。我们首先给出一个合适的斯坦纳点集和一个合适的三角剖分，以此来证明该陈述。接下来，我们将证明对于该特定三角剖分，不会发生翻转事件。

First, we consider the straight skeleton S (P) of P, which has n − 2 inner nodes. Each inner node serves as a Steiner point in our triangulation. Next, we add the arcs of S (P) as diagonals to the triangulation. It remains to properly triangulate the faces of S (P).

首先，我们考虑P的直线骨架S (P)，它有n − 2个内部节点。每个内部节点都作为我们三角剖分中的一个斯坦纳点。接下来，我们将S (P)的弧线作为对角线添加到三角剖分中。剩下的就是正确地对S (P)的面进行三角剖分。

phdthesis_61-70.pdf - 靠岸学术

2.3 a n o v e l w a v e f r o n t - t y p e a p p r o a c h

2.3 一种新型波前方法

2.3.1 Motivation

2.3.1 动机

In the previous section we learned that the use of Steiner vertices can be advantageous in order to reduce the number of flip events in the triangulation-based straight-skeleton algorithm by Aichholzer and Aurenhammer [AA98]. However, the following question remained open: How do we find a good set of Steiner points without knowing the straight skeleton? In the original approach a flip event occurs when a reflex wavefront vertex crosses a triangulation diagonal. These flip events are avoided for the Steiner triangulation presented in the proof of Lemma 2.17, because the reflex wavefront vertices move along triangulation diagonals. The idea is to find a set of Steiner points and a triangulation such that the trajectories of the reflex wavefront vertices are covered by triangulation diagonals.

在前一节中，我们了解到使用斯坦纳顶点可能有利于减少基于三角剖分的直线骨架算法中翻转事件的数量，该算法由Aichholzer和Aurenhammer [AA98]提出。然而，以下问题仍然悬而未决：如何在不知道直线骨架的情况下找到一组好的斯坦纳点？在原始方法中，当一个凹波前顶点穿过三角剖分对角线时，就会发生翻转事件。对于引理2.17的证明中提出的斯坦纳三角剖分，这些翻转事件被避免了，因为凹波前顶点沿着三角剖分对角线移动。其思想是找到一组斯坦纳点和一个三角剖分，使得凹波前顶点的轨迹被三角剖分对角线覆盖。

Assume we are given a non-degenerate polygon P. (Recall, P is non-degenerate if no two motorcycles of M(P) crash simultaneously at the same point.) By Theorem 2.11 we know that the motorcycle graph M(P) covers the reflex arcs of the straight skeleton S (P). We consider the endpoints of the motorcycle traces of M(P) as Steiner points and the traces as diagonals of the triangulation. Theorem 2.11 guarantees that a reflex wavefront vertex does not move beyond its corresponding motorcycle trace. Hence, we obtained a similar situation as in Lemma 2.17, where no flip events are caused by reflex wavefront vertices. In the following we will discuss these two questions: (i) what happens when the wavefront meets a Steiner point and (ii) how do we triangulate the remaining faces?

假设我们给定一个非退化多边形P。（回想一下，如果M(P)的任何两辆摩托车都不会在同一点同时碰撞，则P是非退化的。）根据定理2.11，我们知道摩托车图M(P)覆盖了直线骨架S(P)的凹弧。我们将M(P)的摩托车轨迹的端点视为斯坦纳点，并将这些轨迹视为三角剖分的对角线。定理2.11保证了凹波前顶点不会移动到其对应的摩托车轨迹之外。因此，我们

获得了与引理2.17类似的情况，即没有翻转事件是由凹波前顶点引起的。在下文中，我们将讨论这两个问题：（i）当波前遇到斯坦纳点时会发生什么，以及（ii）我们如何对剩余的面进行三角剖分？

2.3.2 The extended wavefront and a novel straight-skeleton algorithm

2.3.2 扩展波前与一种新颖的直骨架算法

Consider a motorcycle trace that starts at a reflex wavefront vertex v of W (P, 0) and ends at an edge of P at point p. By our construction above, we obtain a Steiner point at p and a triangulation diagonal covering the motorcycle trace that starts at v and ends in p. We define that while the wavefront propagates, the Steiner point at p accompanies the moving intersection of W (P, t) and the motorcycle trace. Metaphorically speaking, the point p “surfs” on the wavefront, along the motorcycle trace, towards v. Analogously, if a motorcycle trace starts at v and crashes into another motorcycle trace at point p then the Steiner point at p keeps still until the wavefront W (P, t) meets p. From that moment, p starts moving on the motorcycle trace towards v. To sum up, we maintain the intersection of the motorcycle graph M(P) and the propagating wavefront W (P, t) and we refer to the additional points as Steiner vertices. More precisely, we call a Steiner vertex, which has not yet been met by the wavefront, a resting Steiner vertex and we call a Steiner vertex, which already moves on the intersection of a motorcycle trace and the wavefront, a moving Steiner vertex. Every resting Steiner vertex lies on the intersection of two motorcycle traces and eventually becomes a moving Steiner vertex when it is reached by the wavefront, see Figure 29.

考虑一个摩托车轨迹，它始于 W (P, 0) 的反射波前顶点 v，并终止于 P 的一条边上的点 p。根据我们上面的构造，我们在 p 处获得一个斯坦纳点，以及一个覆盖摩托车轨迹的三角剖分对角线，该轨迹始于 v，并终止于 p。我们定义，当波前传播时，p 处的斯坦纳点伴随着 W (P, t) 和摩托车轨迹的移动交点。比喻地说，点 p 沿着摩托车轨迹，在波前上“冲浪”，朝着 v 移动。类似地，如果一个摩托车轨迹始于 v 并撞到另一点 p 处的另一个摩托车轨迹，则 p 处的斯坦纳点保持静止，直到波前 W (P, t) 到达 p。从那一刻起，p 开始在摩托车轨迹上朝着 v 移动。总而言之，我们保持摩托车图 M(P) 和传播波前 W (P, t) 的交点，并将这些附加点称为斯坦纳顶点。更准确地说，我们将尚未被波前遇到的斯坦纳顶点称为静止斯坦纳顶点，并将已经在摩托车轨迹和波前交点上移动的斯坦纳顶点称为移动斯坦纳顶点。每个静止斯坦纳顶点都位于两个摩托车轨迹的交点上，并且最终在被波前到达时变为移动斯坦纳顶点，参见图 29。

Definition 2.19 (extended wavefront). Let P be a simple non-degenerate polygon. We denote by M(P, t) those parts of M(P) which have not been swept by W (P, t′) for t′ < t. The extended wavefront W ∗(P, t) is defined as the overlay of W (P, t) and M(P, t) by splitting the edges of W (P, t) at the intersection points accordingly.

定义 2.19 (扩展波前)[NT0]。设 P 为一个简单的非退化多边形。我们用 M(P, t) 表示 M(P) 中未被 W (P, t′) 在 t′ < t 时刻扫过的部分。 extended wavefront W ∗(P, t) 定义为 W (P, t) 和 M(P, t) 的叠加，并在交点处相应地分割 W (P, t) 的边。

Figure 29 illustrates the extended wavefront W ∗(P, t) for a simple polygon P. In the following, we want to remark that P denotes the filled polygon and not only its boundary.

图29展示了一个简单多边形P的扩展波前W ∗(P, t)。在下文中，我们要说明的是，P表示填充的多边形，而不仅仅是它的边界。

Lemma 2.20. We denote by P a simple non-degenerate polygon. Let p be a point in the relative interior of M(P). Then a local disk around p is tessellated into convex slices by M(P).

引理 2.20。我们用 P 表示一个简单的非退化多边形。设 p 为 M(P) 相对内部的一个点。那么，M(P) 将 p 周围的局部圆盘分割成凸切片。

Proof. Since P is non-degenerate it follows that p is also in the relative interior of a motorcycle trace.

证明。由于P是非退化的，因此p也在摩托车轨迹的相对内部。

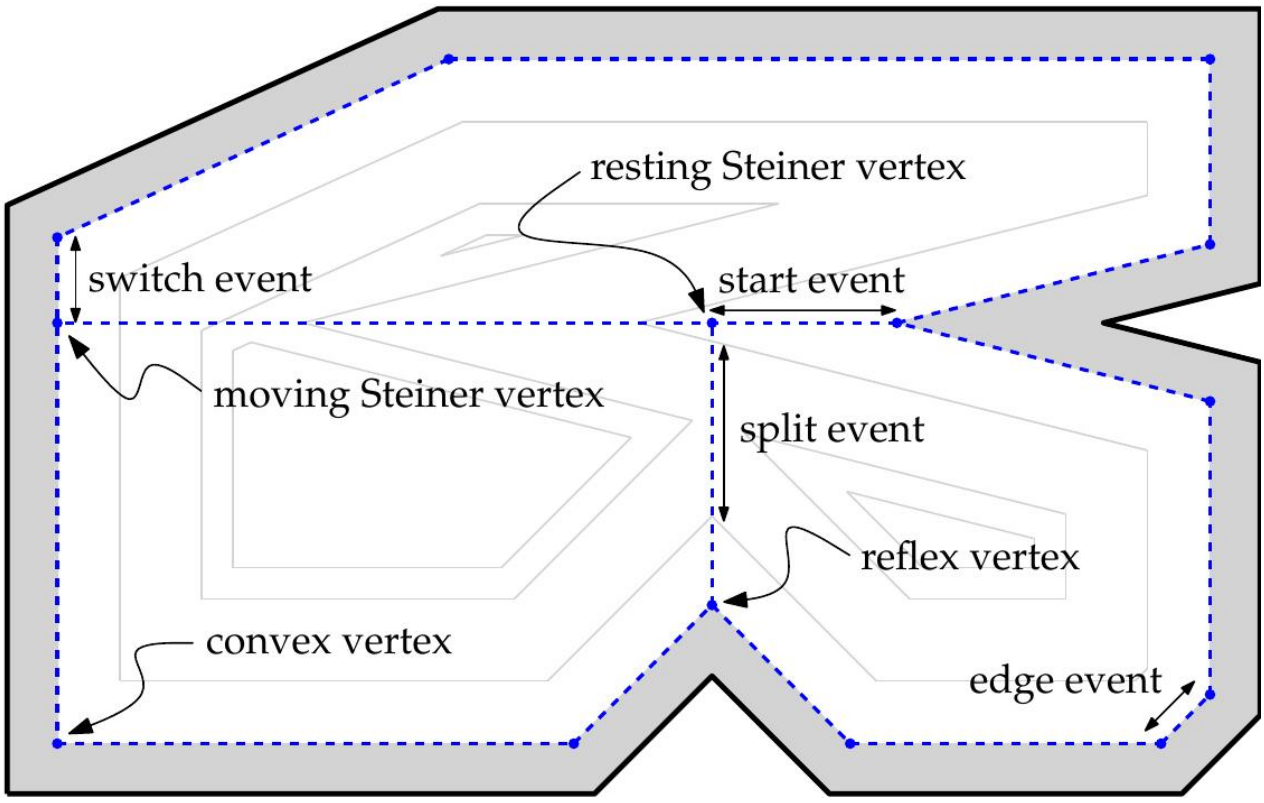


Figure 29: A non-degenerate polygon (bold) and the extended wavefront (dashed) after some time. The area already swept by the wavefront is shaded. The offset curves for three further points in time have been depicted in gray.

图 29：一个非退化多边形（粗体）和经过一段时间后的扩展波前（虚线）。波前已经扫过的区域被阴影覆盖。以灰色描绘了三个后续时间点的偏移曲线。

Lemma 2.21. For any t ≥ 0 the set P \ St′∈[0,t] W ∗(P, t′) consists of open convex faces only.

引理 2.21. 对于任何 t ≥ 0，集合 P \ St′∈[0,t] W ∗(P, t′) 仅由开凸面组成。

Proof. Lemma 2.20 and the fact that at each reflex vertex of P the interior angle is halved by a motorcycle trace, proves the lemma.

证明。引理2.20以及P的每个反射顶点处的内角被摩托车轨迹平分这一事实，证明了该引理。

We now tackle the remaining question concerning the triangulation of the faces induced by the extended wavefront. Recall that we regard the nodes of the motorcycle graph as Steiner vertices and the edges of the motorcycle graph as triangulation diagonals. It follows by Lemma 2.21 that during the propagation of the extended wavefront W ∗(P, t) only neighboring vertices on W ∗(P, t) can meet, since the resulting faces are at any time convex. Hence, we can simply determine all topological changes that occur to W ∗(P, t) by just considering neighboring vertices. In other words, there is no need for a triangulation in order to identify the topological changes.

现在我们来解决关于由扩展波前引起的面进行三角剖分的剩余问题。回想一下，我们将摩托车图的节点视为斯坦纳顶点，并将摩托车图的边视为三角剖分对角线。根据引理 2.21，在扩展波前 W ∗(P, t) 的传播过程中，只有 W ∗(P, t) 上的相邻顶点才能相遇，因为由此产生的面在任何时候都是凸的。因此，我们可以通过仅考虑相邻顶点来简单地确定发生在 W ∗(P, t) 上的所有拓扑变化。换句话说，为了识别拓扑变化，不需要进行三角剖分。
Summarizing, our approach to compute straight skeletons of simple non-degenerate polygons is not to simulate the wavefront W (P, t), but to simulate the extended wavefront W ∗(P, t). As mentioned in Section 1.4.2.1, the difficult problem when simulating the original wavefront W (P, t) is the determination of the split events. In our approach, however, split events are easily determined: A moving Steiner vertex meets a reflex wavefront vertex and both move on the same motorcycle trace. Moreover, every topological change in the extended wavefront W ∗(P, t) is indicated by the collision of two neighboring vertices. In other words, each topological change corresponds to an edge of W ∗(P, t) that collapses to zero length and vice versa. Our algorithm maintains a priority queue Q that contains for each topological change an event. We fetch all events in chronological order and process them. That means, we maintain the extended wavefront depending on the type of event that happened. We distinguish the following types of events. (A detailed discussion is given in Section 2.5.) Note that the different types of events are easily distinguished by the types of vertices involved.

总而言之，我们计算简单非退化多边形直骨架的方法不是模拟波前W (P, t)，而是模拟扩展波前W ∗(P, t)。正如第1.4.2.1节中提到的，在模拟原始波前W (P, t)时，困难的问题是确定分裂事件。然而，在我们的方法中，分裂事件很容易确定：一个移动的斯坦纳顶点遇到一个凹波前顶点，并且两者在同一条摩托车轨迹上移动。此外，扩展波前W ∗(P, t)中的每个拓扑变化都由两个相邻顶点的碰撞指示。换句话说，每个拓扑变化都对应于W ∗(P, t)的一条长度塌陷为零的边，反之亦然。我们的算法维护一个优先级队列Q，其中包含每个拓扑变化的事件。我们按时间顺序获取所有事件并处理它们。这意味着，我们根据发生的事件类型维护扩展波前。我们区分以下几种类型的事件。（详细讨论见第2.5节。）请注意，不同类型的事件很容易通过所涉及的顶点类型来区分。

• (Classical) edge event: Two convex vertices u and v meet. We merge u and v to a single convex vertex and recompute the collapsing times of the two incident edges of $W*(P, t)$. Further, we add the straight-skeleton arcs that have been traced out by u and v to the growing straight-skeleton structure. As a special case, we check whether a whole triangle of $W*(P, t)$ collapsed by this edge event.

• (经典) 边事件：两个凸顶点 u 和 v 相遇。我们将 u 和 v 合并为一个凸顶点，并重新计算 $W*(P, t)$ 的两个相邻边的塌陷时间。此外，我们将由 u 和 v 描绘出的直线骨架弧添加到增长的直线骨架结构中。作为一种特殊情况，我们检查 $W*(P, t)$ 的整个三角形是否因该边事件而塌陷。

• (Classical) split event: A reflex vertex u and a moving Steiner vertex v meet at point p and both move against each other. We add the straight-skeleton arc that has been traced out by u. In general, $W*(P, t)$ is split into two parts. We create corresponding convex vertices that start at p connect them with the resulting parts.

•（经典）分裂事件：一个反射顶点 u 和一个移动的斯坦纳顶点 v 在点 p 相遇，并且都彼此相向移动。我们添加由 u 描绘出的直线骨架弧。通常，$W*(P, t)$ 被分成两个部分。我们创建从 p 开始的相应凸顶点，并将它们与结果部分连接起来。

• Start event: A reflex vertex or a moving Steiner vertex u meets a resting Steiner vertex v. The vertex v becomes a moving Steiner vertex and slides along one incident edge of u. Technically, we have to split an incident edge of u accordingly.

• 开始事件：一个凹顶点或一个移动的斯坦纳顶点u遇到一个静止的斯坦纳顶点v。顶点v变成一个移动的斯坦纳顶点，并沿着u的一个关联边滑动。从技术上讲，我们必须相应地分割u的一个关联边。

• Switch event: A convex vertex u meets a moving Steiner vertex or a reflex Steiner vertex v. The convex vertex u migrates from one convex face of $W*(P, t)$ to a neighboring one, by jumping over v. Technically, the vertex v splits the opposite edge of u and we have to recompute the speed of v.

• 切换事件：一个凸顶点 u 遇到一个移动的斯坦纳顶点或一个凹斯坦纳顶点 v。凸顶点 u 通过跳过 v，从 $W*(P, t)$ 的一个凸面迁移到相邻的一个凸面。从技术上讲，顶点 v 分裂 u 的对边，并且我们必须重新计算 v 的速度。

• All remaining combinations of vertices meeting are guaranteed not to happen. For instance, a convex vertex will never meet a resting Steiner vertex, because a resting Steiner vertex can only have reflex vertices or Steiner vertices as neighbors.

• 保证剩余的顶点组合不会发生。例如，凸顶点永远不会遇到静止的斯坦纳顶点，因为静止的斯坦纳顶点只能有凹顶点或斯坦纳顶点作为邻居。

The correctness of the algorithm follows directly from Lemma 2.21 and Theorem 2.11. The latter guarantees that reflex vertices stay within their corresponding motorcycle traces, i. e. they lead to a split event before they would move beyond the extended

wavefront.

该算法的正确性直接由引理 2.21 和定理 2.11 得出。后者保证了反射顶点停留在其对应的摩托车轨迹内，即，它们在超出扩展波前之前会导致分裂事件。
2.3.3 Runtime analysis and conclusion

2.3.3 运行时分析与结论

Each event involves the recomputation of the collapsing times for a certain amount of edges of the extended wavefront and subsequent modifications in the priority queue Q. Note that each vertex of the extended wavefront has degree two or three. Hence, every event involves only a constant amount of modifications and, thereby, can be handled in O(log n) time.

每个事件都涉及到对扩展波前中一定数量的边的坍塌时间进行重新计算，以及随后对优先级队列Q进行修改。请注意，扩展波前的每个顶点的度数为二或三。因此，每个事件仅涉及常数数量的修改，因此可以在O(log n)时间内处理。

We know that we have O(n) edge events and O(r) split and start events in total, where r ∈ O(n) denotes the number of reflex vertices of P. The number of switch events is bound by O(nr), because a convex vertex can meet a moving Steiner vertex only once. Note that the size of the extended wavefront is linear and the priority queue Q contains at most as many events as the number of edges of the extended wavefront. Hence, our algorithm runs in O(n) space.

我们知道总共有O(n)个边事件和O(r)个分裂和开始事件，其中r ∈ O(n)表示P的反射顶点的数量。切换事件的数量受O(nr)的约束，因为一个凸顶点只能与一个移动的斯坦纳顶点相遇一次。请注意，扩展波前的大小是线性的，并且优先级队列Q最多包含与扩展波前的边数一样多的事件。因此，我们的算法在O(n)空间中运行。

Lemma 2.22. Let P denote a simple non-degenerate polygon with n vertices, where r vertices are reflex. If the motorcycle graph M(P) is known then our algorithm runs in O((n + k) log n) time and O(n) space, where k ∈ O(nr) ⊂ O(n2) denotes the number of switch events that occurred.

引理 2.22. 设 P 表示一个具有 n 个顶点的简单非退化多边形，其中 r 个顶点是凹顶点。如果摩托车图 M(P) 已知，则我们的算法在 O((n + k) log n) 时间和 O(n) 空间内运行，其中 k ∈ O(nr) ⊂ O(n2) 表示发生的切换事件的数量。

For real-world input, it seems very unlikely that Ω(n2) switch events actually occur. This would mean that Ω(n) moving Steiner vertices actually meet with Ω(n) convex vertices.

对于真实世界的输入，Ω(n2)开关事件的实际发生似乎不太可能。这将意味着Ω(n)个移动的斯坦纳顶点实际上与Ω(n)个凸顶点相遇。
This observation is also confirmed by extensive runtime tests in Section 2.5.4. However, a worst-case example, for which Θ(n2) switch events actually occur, can be constructed, see Figure 30. Note that we have to take care that the traces of the motorcycles are

almost parallel within the whole polygon. Furthermore, we require that the trajectories of convex vertices do not intersect before the Ω(n2) switch events occur. That is, the trajectories more or less approach the same point. In other words, this worst-case example is highly contrived.

第2.5.4节中大量的运行时测试也证实了这一观察结果。然而，可以构建一个最坏情况的例子，其中实际发生Θ(n2)次切换事件，参见图30。请注意，我们必须注意摩托车的轨迹在整个多边形内几乎是平行的。此外，我们要求凸顶点的轨迹在Ω(n2)次切换事件发生之前不相交。也就是说，这些轨迹或多或少地接近同一点。换句话说，这个最坏情况的例子是高度人为的。

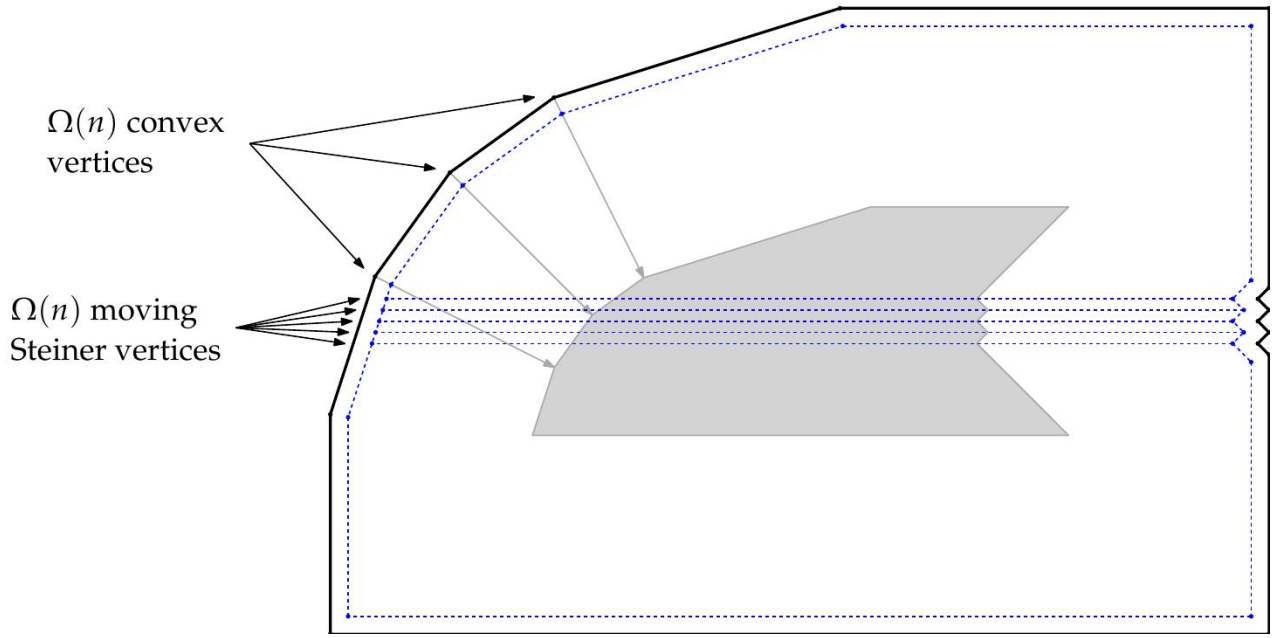


Figure 30: A polygon P for which Ω(n2) switch events occur.

图 30：一个多边形 P，对于该多边形，会发生 Ω(n2) 次开关事件。

The motorcycle graph M(P) of a polygon P with r reflex vertices can be computed in O(r17/11+ϵ) time and space in theory, using the algorithm by Eppstein and Erickson [EE99], cf. Section 1.4.2.3, and in O(r√r log r) time, using the algorithm by Cheng and Vigneron [CV07], cf. Section 1.4.2.4. In practice, the motorcycle graph can be computed in O(r2 log r) time by a straight-forward algorithm enhanced with a priority queue. In Section 3.3, we will present our motorcycle graph implementation Moca, which exhibits an O(r log r) runtime for practical input.

理论上，使用 Eppstein 和 Erickson [EE99] 的算法（参见第 1.4.2.3 节），可以在 O(r17/11+ϵ) 的时间和空间内计算具有 r 个凹顶点的多边形 P 的摩托车图 M(P)；使用 Cheng 和 Vigneron [CV07] 的算法（参见第 1.4.2.4 节），可以在 O(r√r log r) 的时间内计算。在实践中，通过使用优先级队列增强的直接算法，可以在 O(r2 log r) 的时间内计算摩托车图。在第 3.3 节中，我们将介绍我们的摩托车图实现 Moca，它在实际输入中表现出 O(r log r) 的运行时间。

Summarizing, the algorithm presented in this section has a worst-case runtime complexity of O(n2 log n), while it is still easy to implement. This constitutes an improvement by a linear factor compared to the O(n3 log n) worst-case complexity of the algorithm by Aichholzer and Aurenhammer [AA98]. However, their algorithm is capable of computing the straight skeleton of planar straight-line graphs instead of non-degenerate polygons.

总而言之，本节提出的算法的最坏情况运行时间复杂度为 O(n2 log n)，同时它仍然易于实现。与 Aichholzer 和 Aurenhammer [AA98] 的算法的 O(n3 log n) 最坏情况复杂度相比，这是一个线性因子的改进。然而，他们的算法能够计算平面直线图的直线骨架，而不是非退化多边形。

In the following sections we will extend our approach to arbitrary planar straight-line graphs. In the first place this means that we get rid of the non-degeneracy assumption. In order to achieve this we will need to generalize the motorcycle graph accordingly. As a byproduct, this will also result in an alternative characterization of straight skeletons of planar straight-line graphs. Furthermore, the generalized motorcycle graph will allow us to extend the algorithmic approach to planar straight-line graphs.

在接下来的章节中，我们将把我们的方法扩展到任意平面直线图。首先，这意味着我们要摆脱非退化假设。为了实现这一点，我们需要相应地推广摩托车图。作为副产品，这也将产生平面直线图的直线骨架的另一种表征。此外，广义摩托车图将允许我们将算法方法扩展到平面直线图。
2.4 a g e n e r a l i z e d m o t o r c y c l e g r a p h

2.4 广义摩托车图

2.4.1 Motivation and definition

2.4.1 动机与定义

In order to extend the algorithm that was presented in the previous section to arbitrary planar straight-line graphs, we have to generalize the motorcycle graph from simple nondegenerate polygons to arbitrary planar straight-line graphs. While a generalization to nondegenerate planar straight-line graph appears to be straight-forward, the main challenge is to eliminate the need for the assumption of Cheng and Vigneron [CV07]. That is, we need to allow that two motorcycles may crash simultaneously into each other. No matter how the actual generalization is finally realized, we demand the following two properties to hold for the motorcycle graph M(G) of a planar straight-line graph G:

为了将上一节中提出的算法扩展到任意平面直线图，我们必须将摩托车图从简单的非退化多边形推广到任意平面直线图。虽然推广到非退化平面直线图看起来很简单，但主要的挑战是消除对 Cheng 和 Vigneron [CV07] 假设的需求。也就是说，我们需要允许两辆摩托车同时相互碰撞。无论最终如何实现实际的推广，我们都要求对于平面直线图 G 的摩托车图 M(G) 满足以下两个性质：

1. The motorcycle graph M(G) has to cover the reflex arcs of S (G) and

1. 摩托车图M(G)必须覆盖S(G)的自反弧，并且
2. the overlay of the motorcycle graph M(G) and G has to yield a convex tessellation of the plane.

2. 摩托车图M(G)与G的叠加必须产生平面的凸面镶嵌。

Following the definition of the motorcycle graph induced by a simple polygon in Section 1.2.4, we have to specify the motorcycles and the walls that are induced by a given planar straight-line graph G. First of all, we consider the edges of G to be walls. Secondly, we shoot for each reflex wavefront vertex v of W (G, 0) a motorcycle that starts at v and has the same velocity as v. Note that if G forms a polygon then we obtain motorcycles inside and outside of the polygon. Let us consider a planar straight-line graph G, for which a vertex event occurs at point p. This implies that at least two motorcycles m and m′ crashed simultaneously into each other at p. Since a new reflex straight-skeleton arc emanates at p, we need to start a new motorcycle which is going to cover this new straight-skeleton arc. Hence, it will be necessary to generalize the concept of motorcycle graphs such that new motorcycles can emerge at certain moments in time. As a consequence, our generalized motorcycles are specified by a start point, a velocity and, in addition, a start time.

根据第1.2.4节中由简单多边形诱导的摩托车图的定义，我们必须指定由给定的平面直线图G诱导的摩托车和墙。首先，我们认为G的边是墙。其次，对于W (G, 0)的每个凹波前顶点v，我们发射一辆摩托车，该摩托车从v开始，并具有与v相同的速度。请注意，如果G形成一个多边形，那么我们会在多边形的内部和外部获得摩托车。让我们考虑一个平面直线图G，对于该图，顶点事件发生在点p处。这意味着至少有两辆摩托车m和m′在p处同时相互碰撞。由于新的凹直线骨架弧从p发出，我们需要启动一辆新的摩托车，它将覆盖这个新的直线骨架弧。因此，有必要推广摩托车图的概念，以便新的摩托车可以在特定的时间出现。因此，我们的广义摩托车由一个起点、一个速度以及一个起始时间来指定。

For the matter of simplicity we will rephrase the procedure, by which we obtain the set of motorcycles induced by a planar straight-line graph G. Let us denote by e(t) the straightline segment occupied by a wavefront edge e at time t. We demand that two wavefront edges e and e′ of G belong to each motorcycle m and the position of m at time t is given by the intersection e(t) ∩ e′(t). We call the wavefront edge left to the trace of m the left arm of m and the other wavefront edge the right arm of m. Obviously, by specifying the start point and the two arms of a motorcycle m, we implicitly specified the start time of m, too.

为了简便起见，我们将重新措辞获得由平面直线图G诱导的摩托车集合的过程。我们用e(t)表示波前边缘e在时间t所占据的直线段。我们要求G的两个波前边缘e和e′属于每个摩托车m，并且m在时间t的位置由交点e(t) ∩ e′(t)给出。我们将位于m的轨迹左侧的波前边缘称为m的左臂，另一个波前边缘称为m的右臂。显然，通过指定摩托车m的起点和两个臂，我们也隐式地指定了m的开始时间。
For any reflex wavefront vertex v in W (G, 0) we define a motorcycle m that starts at v and whose arms are the two incident wavefront edges of v, see Figure 31 (a). Hence, the motorcycle m has the same velocity as v by construction. Furthermore, we note that each

terminal vertex of G gives rise to two motorcycles by definition, see Figure 31 (b). If two or more motorcycles m1, . . . , mk crash simultaneously at a point p then we consider a local disk D around p. This disk is tessellated into slices by the motorcycle traces. If all slices are convex then we ignore this event. In particular, if another motorcycle reached p earlier, the disk is tessellated into convex slices. In the opposite case, there is exactly one non-convex slice. We may assume that (i) the traces of m1, . . . , mk appear in counter-clockwise order at p and (ii) the non-convex slice is bounded by the traces of m1 and mk. We distinguish the following two cases:

对于W(G, 0)中的任何反射波前顶点v，我们定义一个摩托车m，它从v开始，其臂是v的两个入射波前边，参见图31(a)。因此，根据构造，摩托车m具有与v相同的速度。此外，我们注意到，根据定义，G的每个终端顶点都会产生两辆摩托车，参见图31(b)。如果两辆或多辆摩托车m1, . . . , mk同时在点p处碰撞，那么我们考虑p周围的局部圆盘D。该圆盘被摩托车轨迹分割成片。如果所有切片都是凸的，那么我们忽略此事件。特别地，如果另一辆摩托车更早到达p，则圆盘被分割成凸切片。在相反的情况下，恰好有一个非凸切片。我们可以假设(i)m1, . . . , mk的轨迹以逆时针顺序出现在p处，并且(ii)非凸切片由m1和mk的轨迹界定。我们区分以下两种情况：

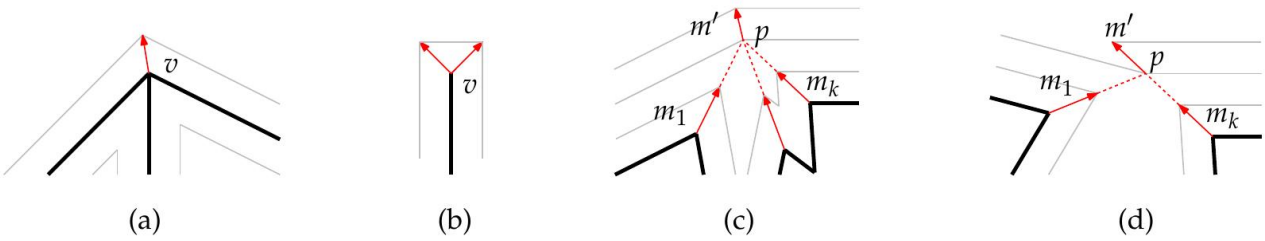


Figure 31: Four different situations for which motorcycles are launched. (a) each reflex wavefront vertex at time zero gives rise to one motorcycle. (b) a terminal vertex gives rise to two motorcycles. (c,d) if two or more motorcycles crash at p such that the traces tessellate a local disc into non-convex slices then we launch a new motorcycle.

图 31：摩托车启动的四种不同情况。(a) 时间零时的每个反射波前顶点产生一辆摩托车。(b) 一个终端顶点产生两辆摩托车。(c,d) 如果两辆或多辆摩托车在 p 处碰撞，使得轨迹将局部圆盘分割成非凸切片，那么我们启动一辆新的摩托车。

1. The left arm of m1 and the right arm of mk span a reflex angle (at the side which has not yet been swept by the wavefront at this time). Then we launch a new motorcycle m′, which starts at p and whose arms are given by the left arm of m1 and the right arm of mk, see Figure 31 (c). This case is necessary in order to cover reflex straight-skeleton arcs that emanate from a vertex event.

1. m1的左臂和mk的右臂跨越一个反射角（在此时尚未被波前扫过的侧面）。然后我们启动一个新的摩托车m′，它从p开始，其臂由m1的左臂和mk的右臂给出，参见图31（c）。为了覆盖从顶点事件发出的反射直骨架弧，这种情况是必要的。

2. The left arm of m1 and the right arm of mk span a convex angle. Then we shoot a new motorcycle m′ which continues the movement of mk. That is, m′ starts at p and has the same pair of arms as mk, see Figure 31 (d).

2. m1的左臂和mk的右臂张成一个凸角。然后我们发射一个新的摩托车m′，它延续mk的运动。也就是说，m′从p开始，并具有与mk相同的一对臂，参见图31（d）。
Letting the motorcycle m′ continue the movement of mk appears to be somewhat arbitrary at the first sight. However, by this definition, we obtain the nice property that the arms of a motorcycle always span a reflex angle at the propagation side. This property, however, turns out to be useful in the proof of Theorem 2.26.

让摩托车m′继续mk的运动，乍一看似乎有些随意。然而，通过这个定义，我们得到了一个很好的性质，即摩托车的臂在传播侧总是跨越一个钝角。然而，这个性质在定理2.26的证明中被证明是有用的。

Also note that Lemma 2.25 becomes trivial for the case illustrated in Figure 31 (d). Indeed, Lemma 2.25 would be also trivial in the case that is illustrated in Figure 31 (c) if we would launch an additional motorcycle that continues the movement of mk. However, it is unclear whether Lemma 2.24 would remain true.

另请注意，引理2.25对于图31(d)中所示的情况变得微不足道。事实上，如果我们启动一辆额外的摩托车来延续mk的运动，引理2.25在图31(c)所示的情况下也会变得微不足道。然而，引理2.24是否仍然成立尚不清楚。

We note that we do not have to simulate the wavefront in order to apply the above case distinction: it suffices to know the propagation direction of the arms of the motorcycles. Furthermore, it could happen that an arm of a motorcycle already vanished in the wavefront propagation process while the motorcycle still moves. The terminology of arms turns out to be advantageous in the subsequent analysis.

我们注意到，为了应用上述情况区分，我们不必模拟波前：知道摩托车臂的传播方向就足够了。此外，有可能在摩托车仍在移动时，摩托车的一个臂已经在波前传播过程中消失。事实证明，臂的术语在后续分析中是有利的。

In both cases of the previous case distinction, we call m1, . . . , mk the ancestors of m′. In particular, we call mk the right-most ancestor and m1 the left-most ancestor. Note that m1 resp. mk can again have ancestors. We define the right-most ancestor chain of m′ as the union of the trace of m′, the trace of mk, the trace of the right-most ancestor of mk and so on. The left-most ancestor chain is defined likewise.

在前述情况区分的两种情况下，我们都称 m1, . . . , mk 为 m′ 的祖先。特别地，我们称 mk 为最右侧的祖先，m1 为最左侧的祖先。注意 m1 可以再次拥有祖先。mk 同样可以再次拥有祖先。我们将 m′ 的最右侧祖先链定义为 m′ 的轨迹、mk 的轨迹、mk 的最右侧祖先的轨迹等等的并集。最左侧祖先链的定义类似。

Definition 2.23 (motorcycle graph induced by a planar straight-line graph). Let G denote a planar straight-line graph. We consider the edges of G as walls and consider the set of motorcycles as elaborated above. The motorcycle graph M(G) induced by G is defined as the arrangement of the resulting motorcycle traces.

定义 2.23 (由平面直线图导出的摩托车图)[NT0]。设 G 表示一个平面直线图。我们将 G 的边视为墙，并考虑如上所述的摩托车集合。由 G 导出的摩托车图 M(G) 定义为所得摩托车轨迹的排列。

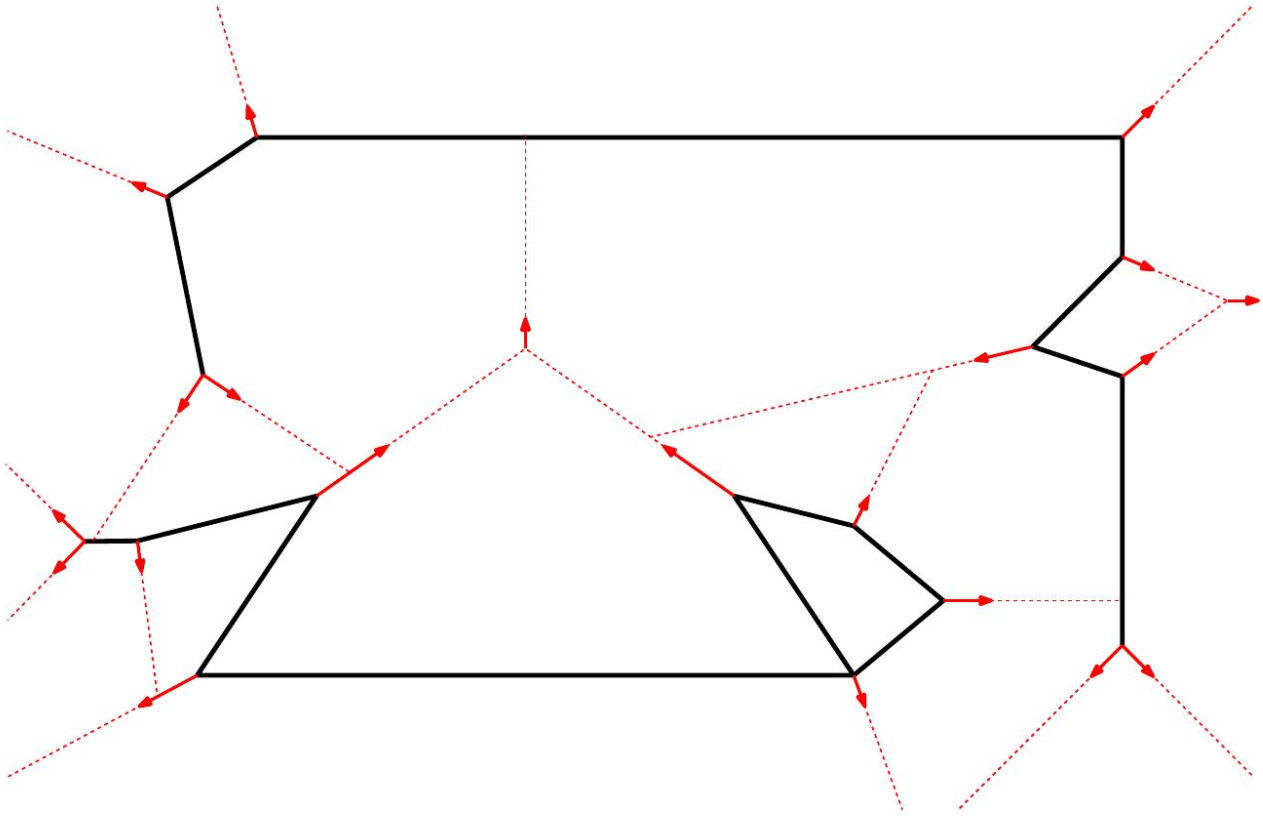


Figure 32: The motorcycle graph M(G), depicted in red, which is induced by the planar straight-line graph G (bold).

图 32：摩托车图 M(G)，以红色描绘，它是由平面直线图 G（粗体）导出的。

In Figure 32 we have illustrated the motorcycle graph M(G) for a planar straight-line graph G. Figure 3 on page 9 illustrates the corresponding straight skeleton S (G).

在图32中，我们展示了平面直线图G的摩托车图M(G)。第9页的图3展示了相应的直线骨架S(G)。

2.4.2 Geometric properties of the generalized motorcycle graph

2.4.2 广义摩托车图的几何性质

Lemma 2.24. The motorcycle graph M(G) of a planar straight-line graph G contains O(n) motorcycle traces.

引理 2.24. 平面直线图 G 的摩托车图 M(G) 包含 O(n) 条摩托车轨迹。

Proof. Let r ∈ O(n) denote the number of reflex wavefront vertices in W (G, 0). Each motorcycle that does not start from a vertex of G starts from the crash point of at least two other motorcycles. Hence, we obtain that M(G) comprises at most 2r − 1 motorcycles traces.

证明。设 r ∈ O(n) 表示 W (G, 0) 中反射波前顶点的数量。每个不是从 G 的顶点开始的摩托车，都是从至少两个其他摩托车的碰撞点开始的。因此，我们得出 M(G) 最多包含 2r − 1 条摩托车轨迹。
Lemma 2.25 ([HH11c]). Let p denote a point in the relative interior of M(G). A local disk around p is tessellated into convex slices by the traces of M(G).

引理 2.25 ([HH11c]).设 p 表示 M(G) 相对内部中的一个点。围绕 p 的局部圆盘被 M(G) 的轨迹细分为凸切片。

Proof. If p is in the relative interior of a trace, the lemma is trivially true. Let us assume that p is the endpoint of k ≥ 2 motorcycle traces. We denote the motorcycles that crashed at p by m1, . . . , mk. Consider a local disk D around p. If D is tessellated into convex slices by the traces of m1, . . . , mk then our assertion holds again. Assume that there is a non-convex slice. We may assume that the motorcycles are numbered in a cyclic order such that (i) their corresponding traces appear counter-clockwise around p and (ii) the trace of m1 and mk bound the reflex slice.

证明。如果 p 位于某个迹的相对内部，则该引理显然成立。让我们假设 p 是 k ≥ 2 个摩托车迹的端点。我们用 m1, . . . , mk 表示在 p 处碰撞的摩托车。考虑 p 周围的局部圆盘 D。如果 D 被 m1, . . . , mk 的迹细分为凸切片，那么我们的断言再次成立。假设存在一个非凸切片。我们可以假设摩托车按循环顺序编号，使得 (i) 它们对应的迹以逆时针方向出现在 p 周围，并且 (ii) m1 和 mk 的迹界定了凹切片。
If the left arm of m1 and the right arm of mk span a convex angle then there is a motorcycle m which continues the movement of mk and the assertion holds again. So, assume that the left arm of m1 and the right arm of mk span a reflex angle. Hence, there is a motorcycle m which started from p and shares its left arm with m1 and its right arm with mk. We have to prove that (i) the traces of m1 and m span a convex angle and (ii) the traces of mk and m span a convex angle. Without loss of generality, we assume that m and mk span an angle greater than π. (The other case is symmetric.) The basic idea is to prove that under this assumption, the existence of m is contradicted because (i) m1 or mk crashes before reaching p or (ii) m1 and mk reach p after another motorcycle was at p before.

如果m1的左臂和mk的右臂构成一个凸角，那么存在一辆摩托车m，它延续了mk的运动，并且该断言再次成立。因此，假设m1的左臂和mk的右臂构成一个优角。因此，存在一辆摩托车m，它从p出发，并且其左臂与m1共享，右臂与mk共享。我们必须证明（i）m1和m的轨迹构成一个凸角，并且（ii）mk和m的轨迹构成一个凸角。不失一般性，我们假设m和mk构成一个大于π的角。（另一种情况是对称的。）基本思想是证明，在这种假设下，m的存在性被反驳，因为（i）m1或mk在到达p之前发生碰撞，或者（ii）m1和mk在另一辆摩托车之前到达p后到达p。

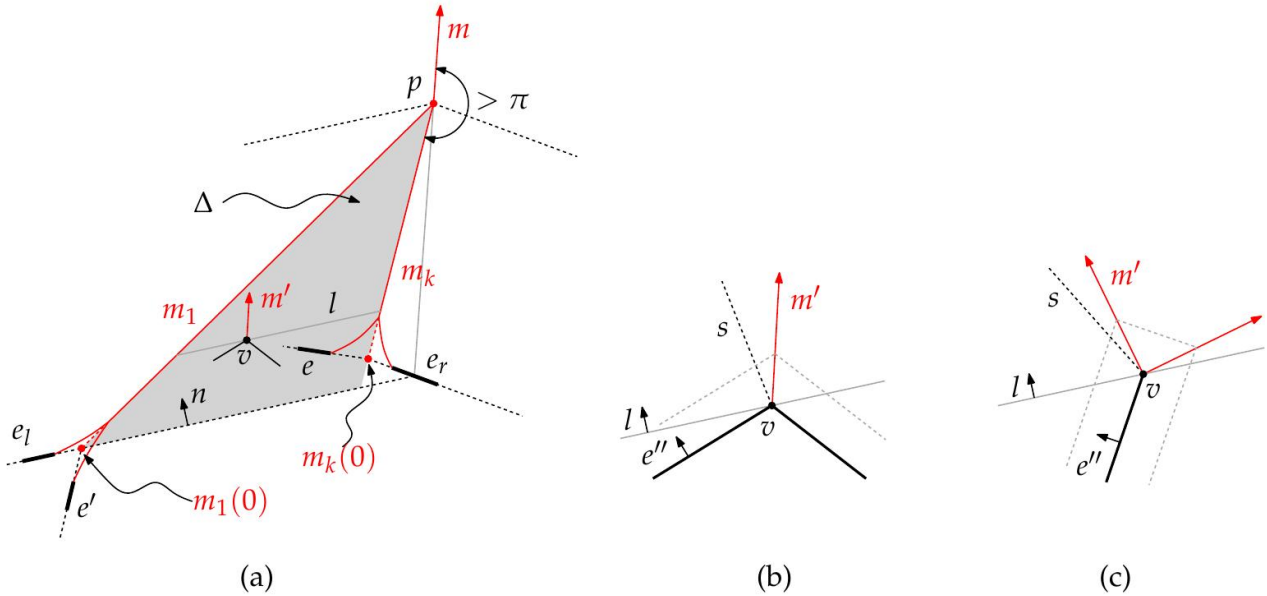


Figure 33: Convex tessellation at simultaneous crashes. (a) if m and mk span a reflex angle then the launch of m is impossible. (b–c) a close-up of the vertex v, where v is (b) a non-terminal vertex and (c) a terminal vertex.

图 33：同时碰撞时的凸镶嵌。(a) 如果 m 和 mk 跨越一个钝角，则 m 的发射是不可能的。(b–c) 顶点 v 的特写，其中 v 是 (b) 非终端顶点，(c) 终端顶点。

We denote by el and er the left and right arm of m, respectively. We note that m shares its right arm with mk and its left arm with m1. Further, we denote by e′ the right arm of m1 and by e the left arm of mk, see Figure 33 (a). Neither m1 nor mk need to start at time zero. We denote by R the right-most ancestor chain of m1, which contains the trace of m1 and ends at an endpoint of e′(0). (Recall that we denote by e(t) the straight-line segment occupied by e at time t.) Likewise, we denote by L the left-most ancestor chain of motorcycle traces of mk.

我们分别用el和er表示m的左臂和右臂。我们注意到m与mk共享其右臂，与m1共享其左臂。此外，我们用e′表示m1的右臂，用e表示mk的左臂，参见图33（a）。m1和mk都不需要在时间零开始。我们用R表示m1的最右侧祖先链，它包含m1的轨迹，并终止于e′(0)的端点。（回想一下，我们用e(t)表示e在时间t占据的直线段。）同样地，我们用L表示mk的摩托车轨迹的最左侧祖先链。

First we show that e(0) is above el(0), i. e., e(0) is on the propagation side of el(0): The point m1(0) must lie on el(0). (If m1 does not start at time zero, we interpret m1(0) as the extrapolation of its movement before its actual starting time.) Likewise, mk(0) must lie on er(0). By assumption, mk(0) lies left to the speed ray of m and right to the speed ray of m1. Hence, the arms of mk span a smaller angle (on the propagation side of the motorcycle) than the one of m. It follows that e(0) lies above el(0).

首先，我们证明 e(0) 位于 el(0) 之上，即 e(0) 位于 el(0): 的传播侧：点 m1(0) 必须位于 el(0) 上。（如果 m1 不是从零时刻开始，我们将 m1(0) 解释为其在实际开始时间之前的运动的外推。）同样，mk(0) 必须位于 er(0) 上。根据假设，mk(0) 位于 m 的速度射线左侧，

且位于 m1 的速度射线右侧。因此，mk 的臂跨越的角度（在摩托车的传播侧）小于 m 的臂跨越的角度。由此得出，e(0) 位于 el(0) 之上。
We denote by ∆ the triangle enclosed by el(0) and the supporting lines of the traces of m1 and mk. Among all vertices of G within ∆, we denote by v a vertex that has maximum orthogonal distance to el(0). Note that at least the endpoints of e(0) are within ∆. The one endpoint that is not incident to L has the greater orthogonal distance of both. We denote by n the propagation vector of el and by l the parallel line of el through v, see Figure 33 (a). The vector n has unit length and is orthogonal to el. If there are multiple vertices of G that lie on l then we may assume that v is the left-most on l (that is, the closest to the trace of m1).

我们用 ∆ 表示由 el(0) 以及 m1 和 mk 的轨迹的支撑线所围成的三角形。在 ∆ 内的 G 的所有顶点中，我们用 v 表示到 el(0) 具有最大正交距离的顶点。请注意，e(0) 的至少一个端点位于 ∆ 内。与 L 不相邻的那个端点具有两者中较大的正交距离。我们用 n 表示 el 的传播向量，用 l 表示穿过 v 的 el 的平行线，参见图 33 (a)。向量 n 具有单位长度并且与 el 正交。如果 G 有多个顶点位于 l 上，那么我们可以假设 v 是 l 上最左边的顶点（即，最靠近 m1) 轨迹的顶点）。

Consider l to be a sweep line moving parallel according to the speed vector n. We show that there is always a motorcycle m′ that emanates from v and which is at any time in front of the sweep line l (or just coinciding). In order to see that we distinguish two cases:

考虑l为一条根据速度矢量n平行移动的扫描线。我们证明，总是存在一辆摩托车m′，它从v发出，并且在任何时候都位于扫描线l的前面（或者仅仅是重合）。为了说明这一点，我们区分两种情况：

• The vertex v has two or more incident edges. Since no incident edge of v lies above l we have a motorcycle m′ emanating from v, see Figure 33 (b). We denote by e″ the left arm of m′. Note that e″(0) is not collinear with l; otherwise there would be another vertex of G on l which is left to v.

• 顶点v有两条或两条以上的入射边。由于v的任何入射边都不在l之上，因此我们有一个从v发出的摩托车m′，参见图33（b）。我们用e″表示m′的左臂。请注意，e″(0)与l不共线；否则，G在l上会有另一个顶点，该顶点位于v的左侧。

We denote by s the bisector of l and e″, which consists of those points that are reached by the propagating supporting line e″(t) and the sweep line l at the same time. Any point right to s is at first reached by e″(t). Since the motorcycle m′ is at any time right to s we see that m′ is always in front of the moving line l.

我们用 s 表示 l 和 e″ 的角平分线，它由传播支撑线 e″(t) 和扫描线 l 同时到达的点组成。任何在 s 右侧的点首先被 e″(t) 到达。由于摩托车 m′ 在任何时候都在 s 的右侧，我们看到 m′ 始终在移动线 l 的前面。

• The vertex v is a terminal vertex. There are two motorcycles emanating from v. We denote by m′ the one whose speed vector has the greater inner product with n, see Figure 33 (c). We denote by e″ the arm of m′ which is parallel to the incident edge of v.

(W.l.o.g. we may assume that this is the left arm. The other case is symmetric.) As in the first case, we denote by s the bisector of the moving line l an e″. Since m′ is right to s (or just overlapping) we see that m′ is never behind the moving line l.
• 顶点v是一个终端顶点。有两辆摩托车从v发出。我们用m′表示速度向量与n有更大内积的那辆摩托车，见图33（c）。我们用e″表示m′的臂，它与v的入射边平行。（不失一般性，我们可以假设这是左臂。另一种情况是对称的。）与第一种情况一样，我们用s表示移动线l和e″的角平分线。由于m′在s的右侧（或只是重叠），我们看到m′永远不会在移动线l的后面。
To sum up, there is always a motorcycle m′ that never falls behind the sweep line l. Next, we note that l intersects R and L in their relative interiors. We conclude the proof with the following case distinction:

总而言之，总是存在一辆摩托车m′，它永远不会落后于清扫线l。接下来，我们注意到l与其相对内部的R和L相交。我们用以下情况区分来结束证明：

1. Assume that m′ reaches R or L. We first see that m1 and mk always stay strictly behind the sweep line l: This is easy to see for m1 since m1(0) starts behind the sweep line l and el propagates with the same speed as l. (Note that m1(t) and el(t) are coinciding.) We assume for a moment that mk would overtake l at some time. Since mk started behind the sweep line l, it follows that mk reaches p before the sweep line does. However, this is a contradiction, because mk cannot reach p before m1 does.

1. 假设 m′ 到达 R 或 L。我们首先看到 m1 和 mk 始终严格地位于扫描线 l 之后：对于 m1 来说，这很容易理解，因为 m1(0) 从扫描线 l 之后开始，并且 el 以与 l 相同的速度传播。（请注意，m1(t) 和 el(t) 是重合的。）我们假设 mk 在某个时刻会超过 l。由于 mk 从扫描线 l 之后开始，因此 mk 在扫描线到达 p 之前到达 p。然而，这是一个矛盾，因为 mk 无法在 m1 到达 p 之前到达 p。

Since m1 and mk stay behind the sweep line l and since m′ is always in front of (or just coinciding with) l, it follows that (i) m1 crashes into m′, or (ii) mk crashes into m′, or (iii) m1 and mk reach p but m′ was at p before. In any case, the launch of m from p is avoided.

由于m1和mk停留在扫描线l之后，并且由于m′始终在l的前面（或恰好与l重合），因此得出结论：（i）m1撞到m′，或者（ii）mk撞到m′，或者（iii）m1和mk到达p，但m′之前已经在p。在任何情况下，都避免了从p发射m。

2. Assume that m′ does not reach R or L. Hence, m′ crashed within Δ. The motorcycle m′ did not crash in a wall. (Otherwise, there would have been a vertex of G within Δ that has a greater orthogonal distance to el(0).) Hence, m′ crashed into the trace of a motorcycle m″, which must have started below l, because no motorcycle could have come through R or L. Hence m″ is faster than m′ w. r. t. direction n. That is, m″ is always in front of the sweep line l after m′ crashed into m″. We consider m″ as the new m′ and apply again our case analysis. Since there is only a finite number of motorcycles, we eventually end up in Case 1 and obtain a contradiction.

2. 假设m′未到达R或L。因此，m′在Δ内坠毁。摩托车m′没有撞到墙。（否则，在Δ内会有一个G的顶点，该顶点到el(0).)的正交距离更大。）因此，m′撞到了一辆摩托车m″的轨迹，该摩托车必定是从l下方开始的，因为没有摩托车能够通过R或L。因此，m″相对于方向n比m′更快。也就是说，在m′撞到m″之后，m″始终位于扫描线l的前面。我们将m″视为新的m′，并再次应用我们的案例分析。由于摩托车的数量是有限的，我们最终会陷入情况1，并得到一个矛盾。

Theorem 2.26 ([HH11c]). The reflex arcs of S (G) are covered by M(G).

定理 2.26 ([HH11c])[NT0]。S (G) 的反射弧被 M(G) 覆盖。

A GENERALIZED MOTORCYCLE GRAPH - 靠岸学术

Proof. The proof consists of two steps. In the first step we state and prove an essential claim, which basically states that the tilted motorcycle traces of M(G) are above the terrain T (G). This proposition is applied later on in a proof by contradiction for the actual theorem.

证明。该证明包含两个步骤。第一步，我们陈述并证明一个重要的声明，该声明基本上指出 M(G) 的倾斜摩托车轨迹位于地形 T(G) 之上。这个命题稍后将在实际定理的反证法中应用。

(1) Let m be a motorcycle and p ∈ R^2 a point on the trace of m. If all valleys of T (G) are covered by tilted motorcycle traces up to the height of mˆ at p then the height of mˆ is greater or equal to the height of T (G) at p. Equality is attained if and only if there is a valley of T (G) that corresponds to mˆ and which exists until p.

（1）设m是一辆摩托车，p ∈ R^2是m轨迹上的一个点。如果T(G)的所有谷都被倾斜的摩托车轨迹覆盖到p处的mˆ高度，那么mˆ的高度大于或等于T(G)在p处的高度。当且仅当存在一个对应于mˆ的T(G)的谷，并且该谷一直存在到p时，等式成立。

We denote by e the right arm of m and we denote by m1, . . . , mk the motorcycles of the right-most ancestor chain of m such that m1(0) is incident to e(0) and mk equals m, see Figure 34 for k = 2. Furthermore, we denote by pi the endpoint of the trace of mi. We arrive at the following observations:

我们用e表示m的右臂，用m1, . . . , mk表示m的最右侧祖先链的摩托车，使得m1(0)与e(0)相连，且mk等于m，对于k = 2，参见图34。此外，我们用pi表示mi的轨迹的端点。我们得出以下观察结果：

• The motorcycles m1, . . . , mk share the same right arm e. As a consequence, the tilted traces mˆ 1, . . . , mˆ k lie on a plane, namely the supporting plane of the terrain face fˆ(e) of the wavefront edge e.

• 摩托车 m1, . . . , mk 共享同一右臂 e。因此，倾斜轨迹 mˆ 1, . . . , mˆ k 位于一个平面上，即波前边缘 e 的地形面 fˆ(e) 的支撑平面。

• The angle between e(0) and the trace of m1 and between the traces of mi and mi+1, with 1 ≤ i ≤ k − 1, are at most 180◦ by Lemma 2.25. However, for any 1 ≤ i ≤ k the motorcycle mi spans with its right arm e an angle of at least 90◦ by definition of the motorcycles.

• 根据引理 2.25，e(0) 与 m1 的迹线之间的夹角，以及 mi 和 mi+1 的迹线之间的夹角（其中 1 ≤ i ≤ k − 1）至多为 180◦。然而，对于任何 1 ≤ i ≤ k，根据摩托车的定义，摩托车 mi 与其右臂 e 张成的角度至少为 90◦。

Let us denote by T the polygonal chain that is defined by the intersection of T (G) with a vertical curtain that is put on the union of the motorcycle traces of m1, . . . , mk. Claim (1) states that the height of mˆ k is greater than or equal to the height of T at p. In order to prove this claim it suffices to show that the slope of T is at any interior point of a trace of mi at most the slope of the tilted trace mˆi. The following proof is an induction-type proof. We show (i) that T is convex within the interior of motorcycle traces, and (ii) that the slope constraint is maintained when migrating from one trace to the next.

令 T 表示由 T (G) 与垂直幕帘的交点定义的多边形链，该垂直幕帘位于 m1, . . . , mk 的摩托车轨迹的并集上。声明 (1) 指出，mˆ k 的高度大于或等于 T 在 p 处的高度。为了证明这个声明，只需证明 T 的斜率在 mi 的轨迹的任何内点处，最多为倾斜轨迹 mˆi 的斜率。以下证明是一个归纳类型的证明。我们证明 (i) T 在摩托车轨迹内部是凸的，以及 (ii) 当从一个轨迹迁移到下一个轨迹时，斜率约束得以保持。

(i) Due to the existence of m1 there is a reflex wavefront vertex that started from m1(0). The wavefront vertex traces a reflex straight skeleton arc and has the same speed as m1 by our definition of the motorcycles. Hence T and mˆ 1 start with the same slope. Let us consider the part T of T that lies above the trace of m1. If there is a reflex vertex in T then we consider the one whose projection q on the plane is closest to m1(0). Obviously there would be a valley of T (G) at q. By assumption there would also be a motorcycle trace covering this valley at q. Since mˆ 1 is above (or just at the same height as) this trace, it follows that m1 would have crashed at q. This is a contradiction. Hence the slope of T is non-increasing above the trace of m1. The same arguments suffice to show that T is non-increasing above the trace of mi if T was below mˆ i at pi−1.

（i）由于存在m1，因此存在一个从m1(0)开始的反射波前顶点。根据我们对摩托车的定义，波前顶点追踪一条反射直线骨架弧，并具有与m1相同的速度。因此，T和mˆ 1以相同的斜率开始。让我们考虑T中位于m1轨迹上方的部分T。如果T中存在一个反射顶点，那么我们考虑其在平面上的投影q最接近m1(0)的那个。显然，在q处会有一个T (G)的谷。根据假设，在q处也会有一条摩托车轨迹覆盖这个谷。由于mˆ 1高于（或至少与）这条轨迹的高度相同，因此m1会在q处发生碰撞。这是一个矛盾。因此，在m1的轨迹上方，T的斜率是非递增的。同样的论证足以表明，如果T在pi−1处低于mˆ i，则T在mi的轨迹上方是非递增的。

(ii) Next, we show that if the slope of T is less than or equal to the slope of mˆ i before pi, then the slope of T is less than or equal to the slope of mˆ i+1 after pi. We denote by e′ the wavefront edge which defines T after pi.

(ii) 接下来，我们证明如果在 pi 之前 T 的斜率小于或等于 mˆ i 的斜率，那么在 pi 之后 T 的斜率小于或等于 mˆ i+1 的斜率。我们用 e′ 表示在 pi 之后定义 T 的波前边缘。
The slope of T before pi can be expressed by the angle between the trace of mi and e′. That is, the slope increases monotonically as the corresponding angle increases. Likewise, we can express the slope of T after pi by the angle between the trace of mi+1 and e′. Moreover, we can express the slopes of mˆ i resp. mˆ i+1 by the angle between mi

and e resp. mi+1 and e. Hence, we can rephrase our assertion: If the angle between mi and e′ is smaller than the angle between mi and e then the angle between mi+1 and e′ is smaller than the angle between mi+1 and e.

在pi之前，T的斜率可以用mi的轨迹和e′之间的夹角来表示。也就是说，斜率随着对应角度的增加而单调增加。同样地，我们可以用mi+1的轨迹和e′之间的夹角来表示pi之后T的斜率。此外，我们可以分别用mˆ i的斜率来表示。mi和e之间以及mi+1和e之间的夹角分别表示mˆ i+1。因此，我们可以这样复述我们的断言：如果mi和e′之间的夹角小于mi和e之间的夹角，那么mi+1和e′之间的夹角小于mi+1和e之间的夹角。

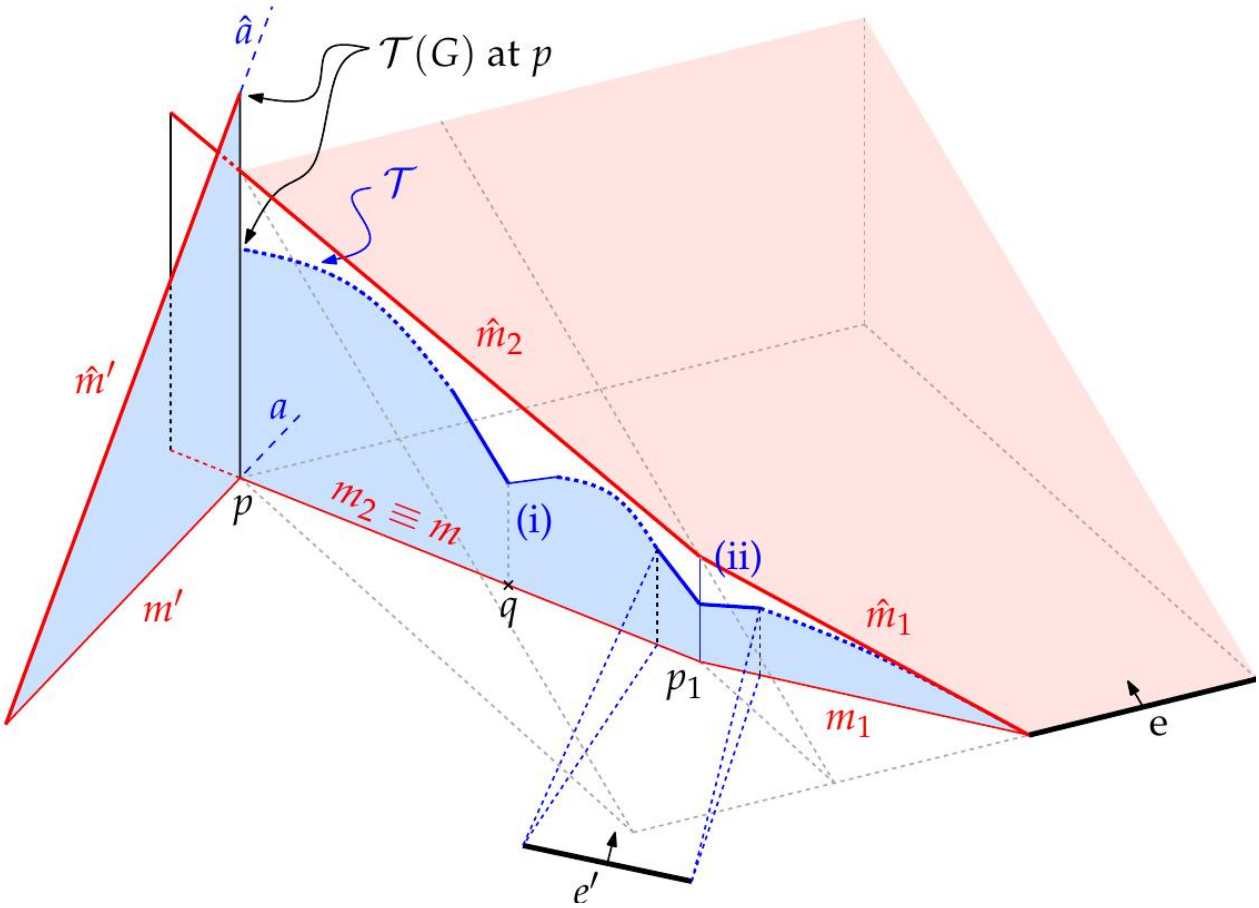


Figure 34: Proof of Theorem 2.26. The terrain T (G) is always below the tilted motorcycle traces mˆ k. The proof shows that the slope of the terrain (blue) is at any point at most the slope of the tilted motorcycle traces (red) since the situations (i) and (ii) (shown in solid red and described in the proof) do not occur.

图 34：定理 2.26 的证明。地形 T(G) 始终低于倾斜的摩托车轨迹 mˆ k。该证明表明，地形（蓝色）的斜率在任何点都至多是倾斜的摩托车轨迹（红色）的斜率，因为情况 (i) 和 (ii)（以红色实线显示并在证明中描述）不会发生。

Let us consider Figure 35. We denote by l the bisector between e and e′ on the left side and by r the bisector on the right side. Hence, we have to prove that mi+1 lies right to l and left to r. Our premise states that the angle between e′ and mi is less than or equal to the angle between e and mi. We denote by el the left arm of mi and by ti the time when e reaches pi. Assume that we rotate e′ counter-clockwise around pi until e′ is parallel with e(ti). Then l is falling onto e(ti) and r is perpendicular to e(ti). Vice versa, assume that we

rotate e′ clock-wise around pi until e′ is parallel with el. Then l is on the supporting line of mi and r is on the bisector of mi and e(ti). We shaded the valid domains for l and r in Figure 35 accordingly. Since the angle between mi and mi+1 is convex, mi+1 is right to the domain of l. Since mi+1 and e enclose an angle of at least 90◦, mi+1 is left to the domain of r. Summarizing, for every position of e′, which conforms to our initial assumption, mi+1 encloses a smaller angle with e′ than with e.

让我们考虑图 35。我们用 l 表示 e 和 e′ 左侧的角平分线，用 r 表示右侧的角平分线。因此，我们必须证明 mi+1 位于 l 的右侧和 r 的左侧。我们的前提是 e′ 和 mi 之间的角度小于或等于 e 和 mi 之间的角度。我们用 el 表示 mi 的左臂，用 ti 表示 e 到达 pi 的时间。假设我们围绕 pi 逆时针旋转 e′，直到 e′ 与 e(ti) 平行。那么 l 将落在 e(ti) 上，r 将垂直于 e(ti)。反之，假设我们围绕 pi 顺时针旋转 e′，直到 e′ 与 el 平行。那么 l 位于 mi 的支撑线上，r 位于 mi 和 e(ti) 的角平分线上。我们在图 35 中相应地标出了 l 和 r 的有效区域。由于 mi 和 mi+1 之间的角度是凸的，因此 mi+1 位于 l 区域的右侧。由于 mi+1 和 e 之间的夹角至少为 90◦，因此 mi+1 位于 r 区域的左侧。总而言之，对于 e′ 的每个符合我们初始假设的位置，mi+1 与 e′ 之间的夹角小于与 e 之间的夹角。

Combining arguments (i) and (ii) yields an induction-type proof for Claim (1). Basically, we showed that the distance between T and the tilted motorcycle traces is (not necessarily strictly) monotonically increasing. If T and the tilted motorcycle traces are overlapping until p then equality for the height of mˆ and T (G) at p is attained. If T leaves the tilted motorcycle traces at some point then T (G) is strictly below mˆ at p.

结合论证 (i) 和 (ii) 得出对声明 (1) 的归纳式证明。基本上，我们证明了 T 和倾斜的摩托车轨迹之间的距离是（不一定严格地）单调递增的。如果 T 和倾斜的摩托车轨迹重叠到 p，那么 mˆ 和 T (G) 在 p 处的高度相等。如果 T 在某个点离开倾斜的摩托车轨迹，那么 T (G) 在 p 处严格低于 mˆ。

We now return our attention to Figure 34 and use Claim (1) in a proof by contradiction of Theorem 2.26. Assume that there is a reflex arc a in S (G) that is only partially covered by a motorcycle trace m′. Hence, m′ crashed into a motorcycle m. We denote by p the crashing point of m′. Without loss of generality we assume that the height of mˆ ′ at p is lowest. (By this assumption we can assume that a is at least partially covered. If a would not be covered at all, then a was not incident to G and at least one of its reflex ancestor arcs was not covered completely.) Hence, all valleys of T (G) are covered by motorcycle traces up to the height of mˆ ′ at p. By Claim (1) we know that T (G) is below mˆ at p. On the other hand, we know that T (G) has the same height as mˆ ′ at p. (See the left side of Figure 34.) This contradiction finally concludes the proof.

现在我们将注意力转回图34，并使用引理(1)通过反证法证明定理2.26。假设在S(G)中存在一个仅被摩托车轨迹m′部分覆盖的反射弧a。因此，m′撞上了一辆摩托车m。我们用p表示m′的碰撞点。不失一般性，我们假设mˆ′在p处的高度是最低的。（通过这个假设，我们可以假设a至少被部分覆盖。如果a根本没有被覆盖，那么a就不会与G相关联，并且其至少一个反射祖先弧没有被完全覆盖。）因此，T(G)的所有谷都被摩托车轨迹覆盖到mˆ′在p处的高度。根据引理(1)，我们知道T(G)在p处低于mˆ。另一方面，我们知道T(G)在′处与mˆp具有相同的高度。（参见图34的左侧。）这个矛盾最终结束了证明。

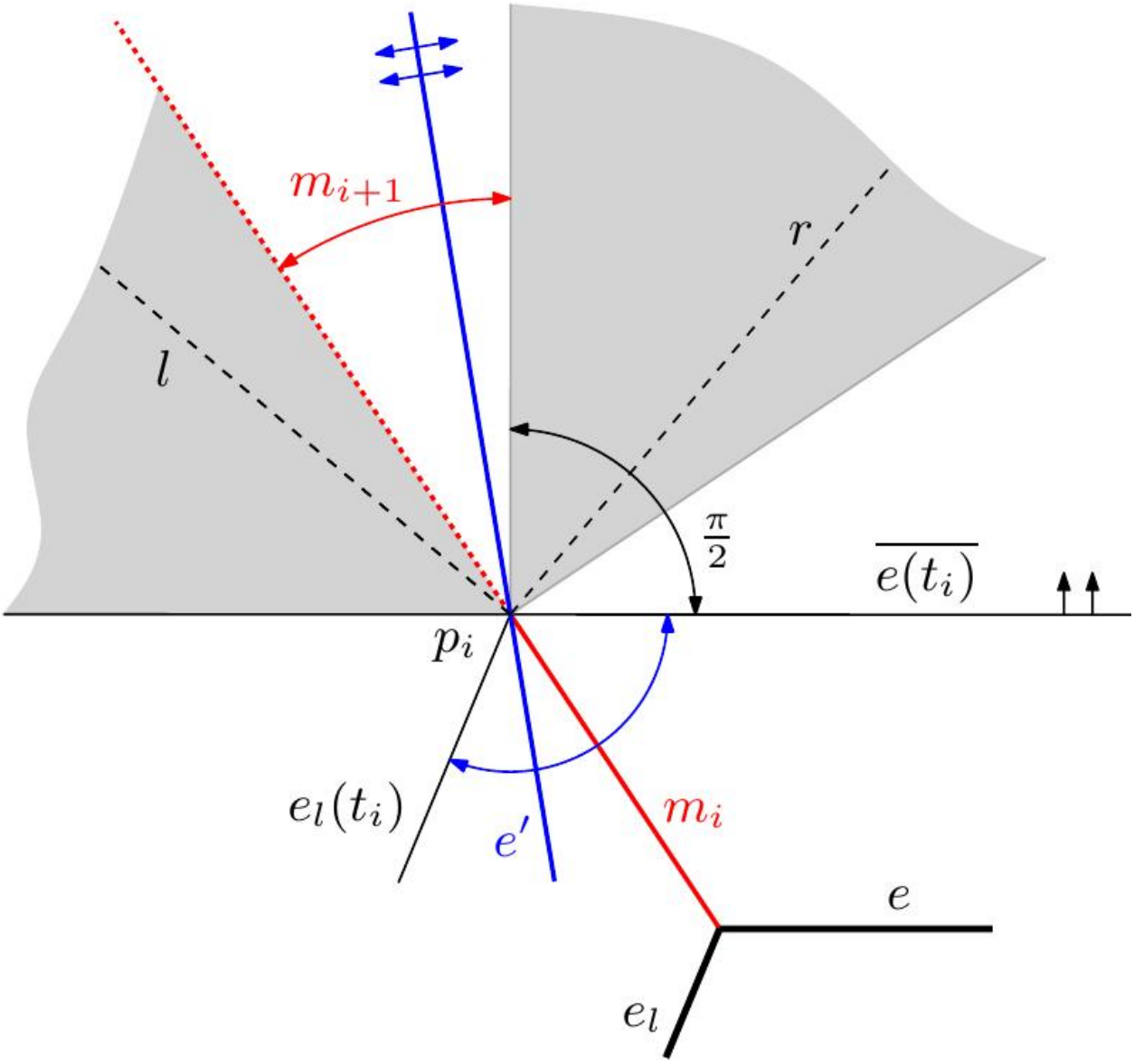


Figure 35: Proof of Theorem 2.26, case (ii). The shaded areas depict the valid domains for l and r and mi+1 is at any time right to l and left to r.

图 35：定理 2.26 的证明，情况 (ii)。阴影区域描绘了 l 和 r 的有效域，并且 mi+1 在任何时候都位于 l 的右侧和 r 的左侧。

Theorem 2.26 extends Theorem 2.11 by Cheng and Vigneron [CV07]. In their proof they assumed that no two motorcycles crashed into each other. The idea of their proof is incorporated into the proof of Case (i) of Claim (1). Claim (1) is a useful tool for its own, so we can cast it to the following corollary.

定理 2.26 扩展了 Cheng 和 Vigneron [CV07] 的定理 2.11。在他们的证明中，他们假设没有两辆摩托车相互碰撞。他们的证明思想被纳入了声明 (1) 中情况 (i) 的证明。声明 (1) 本身就是一个有用的工具，因此我们可以将其转化为以下推论。

Corollary 2.27. Let m be a motorcycle of M(G) and p $\in R^2$ a point on the trace of m. The height of mˆ is greater or equal to the height of T (G) at p. Equality is attained if and only if the valley of T (G), which corresponds to m, exists until p.

推论 2.27. 设 m 为 M(G) 的一个摩托车，且 p ∈ R^2 为 m 轨迹上的一个点。mˆ 的高度大于或等于 T (G) 在 p 处的高度。当且仅当对应于 m 的 T (G) 的谷存在直到 p 时，等式成立。
2.4.3 The lower envelope based on the generalized motorcycle graph

2.4.3 基于广义摩托车图的下包络线

Let us revisit the discussion concerning the alternative characterization of the straight skeleton S (G), by using a lower-envelope representation of the terrain T (G) from Section 2.1. Recall that Theorem 2.7 by Eppstein and Erickson [EE99] provides a lower-envelope representation for T (G), but their slabs depend on the length of the reflex arcs of the straight skeleton. On the other hand, Theorem 2.14 by Cheng and Vigneron [CV07] provides a lower-envelope representation of T (P) that does not depend on the straight skeleton, but their representation can only be applied to simple non-degenerate polygons P with holes.

让我们重新审视关于直线骨架S(G)的另一种表征的讨论，通过使用第2.1节中地形T(G)的下包络表示。回想一下，Eppstein和Erickson [EE99]的定理2.7提供了T(G)的下包络表示，但他们的分片取决于直线骨架的凹弧的长度。另一方面，Cheng和Vigneron [CV07]的定理2.14提供了T(P)的下包络表示，该表示不依赖于直线骨架，但他们的表示只能应用于带有孔的简单非退化多边形P。

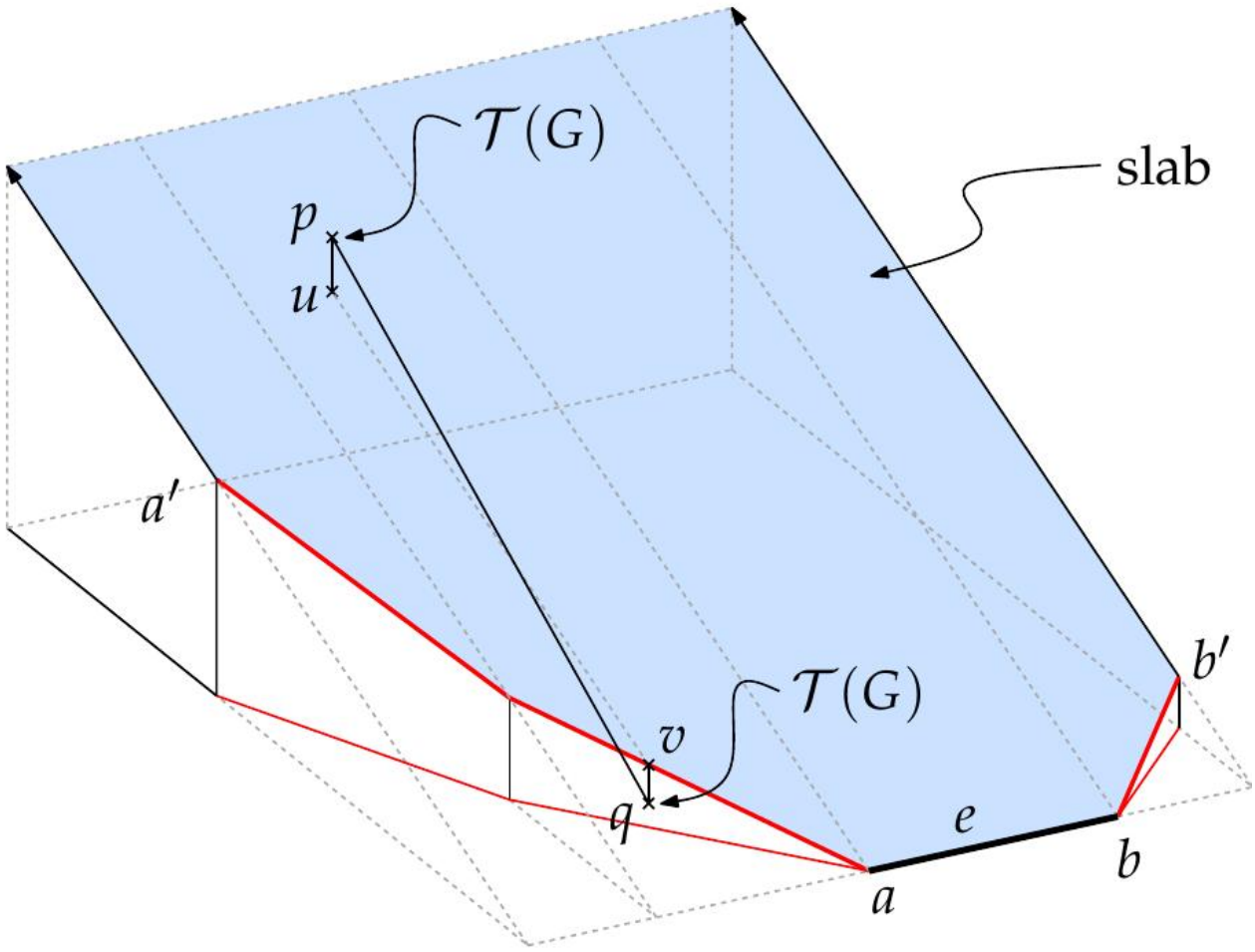


Figure 36: The shaded area illustrates the slab defined by the wavefront edge e. The slab lies on the supporting plane of fˆ(e) and is bounded from below by the tilted motorcycle traces that have e as an arm.

图 36：阴影区域表示由波前边缘 e 定义的板。该板位于 f^(e) 的支撑平面上，并以下臂为 e 的倾斜摩托车轨迹为下界。
Summarizing, no lower-envelope representation is known for the terrain T (G) of planar straight-line graphs G, which does not depend on the straight skeleton.

总而言之，对于平面直线图G的地形T(G)，目前尚不清楚是否存在不依赖于直线骨架的下包络表示。

In the following we extend the slab construction, which is based on the motorcycle graph, to the generalized motorcycle graph, see Figure 36.

在下文中，我们将基于摩托车图的平板构造扩展到广义摩托车图，参见图 36。

Definition 2.28 (lower envelope). Let e denote a wavefront edge of a planar straight-line graph G and let a and b denote the endpoints of e(0). If there is a motorcycle that starts at a and has the right arm e then we consider the whole chain of tilted motorcycles traces, starting at a and ending at a′, whose right arms are e. If there is no such motorcycle then we define a′ := a. Likewise we consider the chain of tilted motorcycle traces starting at b and ending at b′ whose left arms are e. We define a slab for each e that is contained in the supporting plane of the terrain face f^(e) and which is bounded from below by e and the motorcycle traces mentioned above. At the endpoints a′ and b′ the slab is bounded by rays which are perpendicular to e(0). We denote by L(G) the lower envelope of the these slabs.

定义 2.28 (下包络线).设 e 表示平面直线图 G 的波前边缘，并设 a 和 b 表示 e(0) 的端点。如果存在一辆从 a 出发且具有右臂 e 的摩托车，那么我们考虑以 a 开始，以 a′ 结束的倾斜摩托车轨迹的整个链条，其右臂为 e。如果没有这样的摩托车，那么我们定义 a′ := a。同样地，我们考虑以 b 开始，以 b′ 结束的倾斜摩托车轨迹的链条，其左臂为 e。我们为每个 e 定义一个板，该板包含在地形面 f^(e) 的支撑平面中，并且从下方由 e 和上述摩托车轨迹限定。在端点 a′ 和 b′ 处，该板由垂直于 e(0) 的射线限定。我们用 L(G) 表示这些板的下包络线。

Theorem 2.29 ([HH11c]). The lower envelope L(G) is identical to T (G).

定理 2.29 ([HH11c]).下包络 L(G) 与 T(G) 相同。

Proof. Each face f^(e) of T (G) is contained in the corresponding slab of e. It follows that no point of L(G) is above T (G) and it remains to show that no point of T (G) is above L(G). Assume, to the contrary, that a point p ∈ T (G) is above a slab of an edge e, see Figure 36. We project p down to the slab and denote the projection point by u. Then we project u down along the steepest descent of the slab until we hit e or one of the tilted motorcycle traces and obtain the point v. If v is on a tilted trace then Corollary 2.27 implies that we can project v down to T (G) and obtain a point q. If v is on e(0), we set q := v. Since the line between u and v has slope 1, the slope of pq is greater than 1. This is a contradiction to Lemma 2.1.

证明。T(G) 的每个面 f^(e) 都包含在 e 的相应板中。由此可见，L(G) 中没有点在 T(G) 之上，因此只需证明 T(G) 中没有点在 L(G) 之上。反之，假设点 p ∈ T(G) 在边 e 的板之上，参见图 [NT3]36]。我们将 p 向下投影到板上，并将投影点表示为 u。然后，我们沿着板的最陡下降方向将 u 向下投影，直到我们到达 e 或倾斜的摩托车轨迹之一，并获得点 v。如果 v 在倾斜的轨迹上，则推论 2.27 意味着我们可以将 v 向下投影到 T(G) 并获得点 q。如果 v 在 e(0) 上，我们设置 q := v。由于 u 和 v 之间的线的斜率为 1，因此 pq 的斜率大于 1。这与引理 2.1 相矛盾。
As mentioned above, Theorem 2.29 extends Theorem 2.14 by Cheng and Vigneron [CV07] to planar straight-line graphs G. However, Theorem 2.29 also extends Theorem 2.7 by Eppstein and Erickson [EE99] since L(G) is based on M(G) and does not depend on the length of the reflex arcs of S (G). This difference, however, is essential when we attempt to compute S (G) via a lower-envelope computation, see below. Besides, Theorem 2.29 provides an alternative, non-procedural way to define S (G) as the lower envelope of partially linear functions, as discussed in Section 2.1.

如上所述，定理2.29将Cheng和Vigneron [CV07]的定理2.14扩展到平面直线图G。然而，定理2.29也扩展了Eppstein和Erickson [EE99]的定理2.7，因为L(G)基于M(G)，并且不依赖于S(G)的反射弧的长度。然而，当我们尝试通过下包络计算来计算S(G)时，这种差异至关重要，见下文。此外，定理2.29提供了一种替代的、非过程化的方法来定义S(G)，即作为部分线性函数的下包络，如第2.1节中所讨论的。

computing S (G) using graphics hardware Theorem 2.29 admits a simple method to render T (G) without the knowledge of S (G). One first computes the motorcycle graph M(G) by a conventional algorithm (on the CPU) and then constructs the slabs as illustrated in Figure 36. We paint each slab with a different color. By rendering the set of slabs, while looking at them from below, one obtains an image which shows L(G). Hence, by employing techniques described by Hoff et al. [HCK+99], one can compute the straight skeleton of planar straight line graphs using graphics hardware. The idea is that each face f^(e) corresponds to a set of pixels of the same color.

使用图形硬件计算 S (G) 定理 2.29 允许使用一种简单的方法来渲染 T (G)，而无需了解 S (G)。首先通过传统算法（在 CPU 上）计算摩托车图 M(G)，然后如图 36 所示构建切片。我们用不同的颜色绘制每个切片。通过渲染这组切片，同时从下方观察它们，可以获得显示 L(G) 的图像。因此，通过采用 Hoff 等人 [HCK+99] 描述的技术，可以使用图形硬件计算平面直线图的直线骨架。其思想是每个面 f^(e) 对应于一组相同颜色的像素。

2.5 t h e g e n e r a l w a v e f r o n t - t y p e a l g o r i t h m

2.5 通用波前类型算法

2.5.1 Details of the general algorithm

2.5.1 通用算法的细节

Let us recall the basic building blocks of the simple algorithm of Section 2.3. Firstly, we require that the motorcycle graph covers the reflex arcs of the straight skeleton, which is guaranteed by Theorem 2.26. Secondly, we require that the overlay of the motorcycle

graph and the input graph results in a convex tessellation of the plane.

让我们回顾一下第2.3节简单算法的基本组成部分。首先，我们要求摩托车图覆盖直线骨架的反射弧，这由定理2.26保证。其次，我们要求摩托车图和输入图的叠加导致平面的凸面镶嵌。
Definition 2.30 (extended wavefront). Let G denote a planar straight-line graph. We define by M(G, t) those parts of M(G) which have not been swept by W (G, t′) for t′ < t. The extended wavefront W ∗(G, t) is defined as the overlay of W (G, t) and M(G, t) by splitting the edges of W (G, t) at the intersection points accordingly.

定义 2.30（扩展波前）[NT0]。令 G 表示一个平面直线图。我们用 M(G, t) 表示 M(G) 中未被 W (G, t′) 在 t′ < t 时刻扫过的部分。extended wavefront W ∗(G, t) 定义为 W (G, t) 和 M(G, t) 的叠加，相应地在交点处分割 W (G, t) 的边。

Lemma 2.31. For any t ≥ 0 the set R2 \ St′∈[0,t] W ∗(G, t′) consists of open convex faces only.

引理 2.31. 对于任何 t ≥ 0，集合 R2 \ St′∈[0,t] W ∗(G, t′) 仅由开放的凸面组成。

Proof. The assertion follows directly from Lemma 2.25 and the fact that each reflex angle at a reflex wavefront vertex is split by a motorcycle trace.

证明。该论断直接由引理2.25以及每个反射波前顶点处的反射角都被摩托车轨迹分割这一事实得出。

We adopt the terms resting Steiner vertex and moving Steiner vertex from Section 2.3. In addition to these terms we refer to a vertex of W ∗(G, t) that marks the crash of two or more motorcycles as multi Steiner vertex, see Figure 37.

我们采用第2.3节中的术语：静止斯坦纳顶点和移动斯坦纳顶点。除了这些术语之外，我们将W ∗(G, t)的顶点，该顶点标志着两辆或多辆摩托车碰撞的点，称为多重斯坦纳顶点，参见图37。
The basic algorithm from Section 2.3 remains the same. Lemma 2.31 guarantees that any topological change in the extended wavefront is indicated by the collapse of an edge of the extended wavefront to zero length. We start with the initial extended wavefront W ∗(G, 0). For each edge e of W ∗(G, 0) we insert an event into a priority queue Q if the collapsing time of e is positive. The events are prioritized by their time of occurrence. After the initialization we fetch from Q one event after the other and process it. That is, for each event we apply local modifications of the extended wavefront and maintain the priority queue Q accordingly. (Note that if we use a maximizing heap as the underlying data structure for Q we may also remove any element in O(log n) time if we already have a pointer to the element.) In the following we are discussing the different types of events.

第2.3节中的基本算法保持不变。引理2.31保证了扩展波前的任何拓扑变化都由扩展波前的一条边塌缩为零长度来指示。我们从初始扩展波前W ∗(G, 0)开始。对于W ∗(G, 0)的每条边e，如果e的塌缩时间为正，则我们将一个事件插入到优先级队列Q中。事件按其发生时间排序。初始化后，我们从Q中逐个获取事件并处理它。也就是说，对于每个事件，我们

应用扩展波前的局部修改，并相应地维护优先级队列Q。（请注意，如果我们使用最大堆作为Q的底层数据结构，如果已经有一个指向该元素的指针，我们也可以在O(log n)时间内删除任何元素。）下面我们将讨论不同类型的事件。

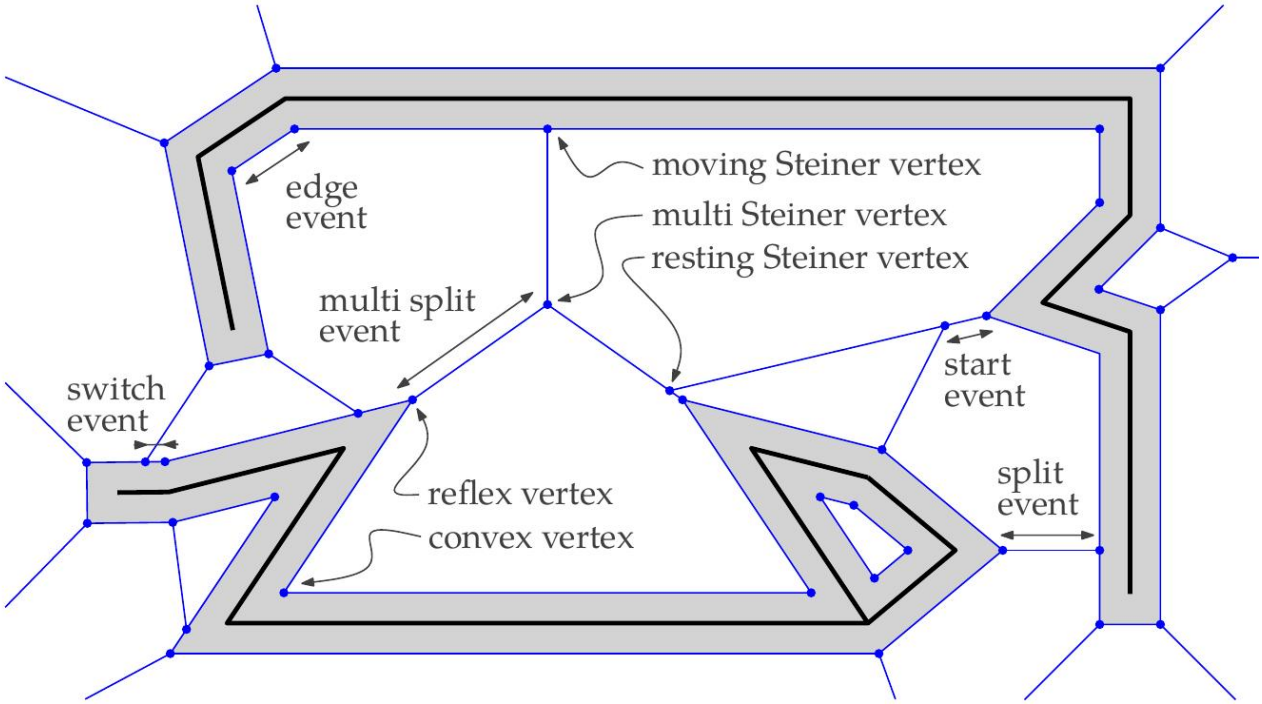


Figure 37: A planar straight-line graph (bold) and the extended wavefront (dashed) after some time. The area already swept by the wavefront is shaded.

图 37：一个平面直线图（粗体）和一段时间后的扩展波前（虚线）。波前已经扫过的区域被阴影覆盖。

• (Classical) edge event: The edge e between two convex vertices u and v collapsed, see Figure 38 (a). We remove e from the graph and merge the vertices u and v to a single convex vertex w. The two edges incident to w determine the new velocity of w. Consequently, we need to recompute the collapsing times of the two incident edges of w and adapt the entries in Q.

•（经典）边事件：两个凸顶点 u 和 v 之间的边 e 坍塌，参见图 38 (a)。我们从图中移除 e，并将顶点 u 和 v 合并为单个凸顶点 w。与 w 相邻的两条边决定了 w 的新速度。因此，我们需要重新计算 w 的两条相邻边的坍塌时间，并调整 Q 中的条目。

Finally, we add the two convex straight-skeleton arcs that were traced out by u and v to the straight-skeleton graph. Altering O(1) entries in Q costs us O(log n) time.

最后，我们将由u和v描绘出的两个凸直线骨架弧添加到直线骨架图中。更改Q中的O(1)个条目需要花费我们O(log n)的时间。

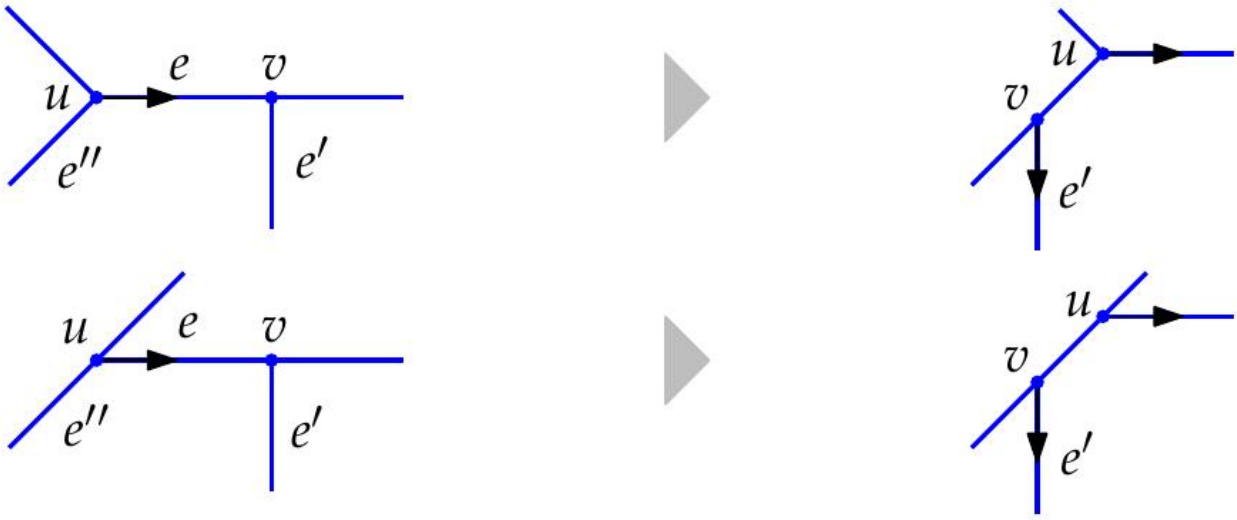
• (Classical) split event: The edge e between a reflex vertex u and a moving Steiner vertex v collapsed, see Figure 38 (b). Note that v is the Steiner vertex that corresponds to the crash of the motorcycle whose trace covers the reflex arc that is traced out by u. We denote by ul resp. vl the two vertices which are incident to u resp. v and left to the

trajectory of u. The vertices ur, vr are denoted likewise. We remove the edge e and merge the vertices ul and vl to a new convex vertex wl. Its velocity is determined by the two incident edges. As a special case we check whether the two incident edges of wl are parallel. This means that the entire convex face of $W*(G, t)$, to which wl belongs, collapsed at the time when the split event occurred. The right side, with the vertices ur and vr, is processed likewise.
• (经典) 分裂事件：反射顶点 u 和移动的斯坦纳顶点 v 之间的边 e 坍缩，参见图 38 (b)。请注意，v 是对应于摩托车碰撞的斯坦纳顶点，该摩托车的轨迹覆盖了由 u 描绘出的反射弧。我们用 ul 和 vl 分别表示与 u 和 v 相邻且位于 u 轨迹左侧的两个顶点。顶点 ur, vr 的表示方式类似。我们移除边 e，并将顶点 ul 和 vl 合并为一个新的凸顶点 wl。其速度由两条相邻的边决定。作为一种特殊情况，我们检查 wl 的两条相邻边是否平行。这意味着 $W*(G, t)$ 的整个凸面（wl 属于该凸面）在分裂事件发生时坍缩。右侧，包含顶点 ur 和 vr，也以类似的方式处理。
Since u crashed at this split event, we need to add the reflex arc that is traced out by u, to our straight-skeleton structure. Altering O(1) entries in Q costs us again O(log n) time.

由于u在此分割事件中崩溃，我们需要将由u追踪出的反射弧添加到我们的直线骨架结构中。在Q中更改O(1)个条目再次花费我们O(log n)的时间。

(a) edge event

(b) split event



(c) start event

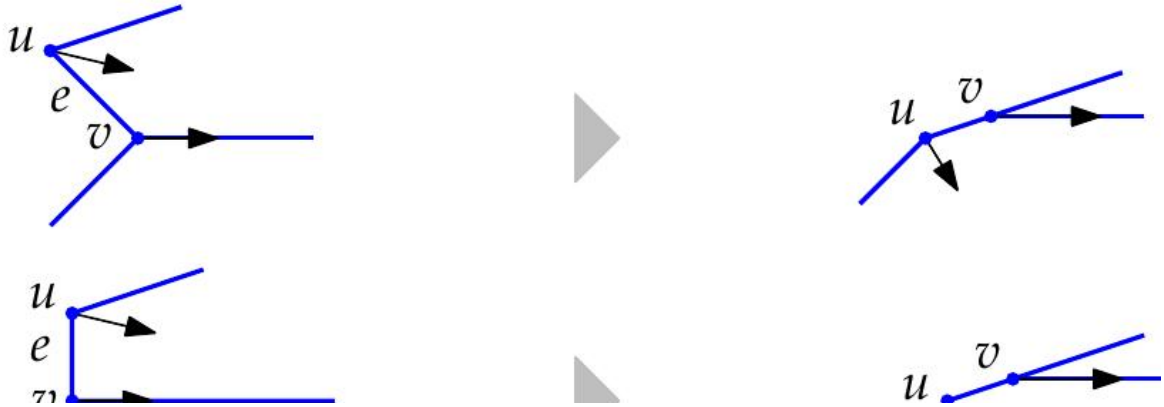


Figure 38: Different types of events that occur during the propagation of the extended wavefront.

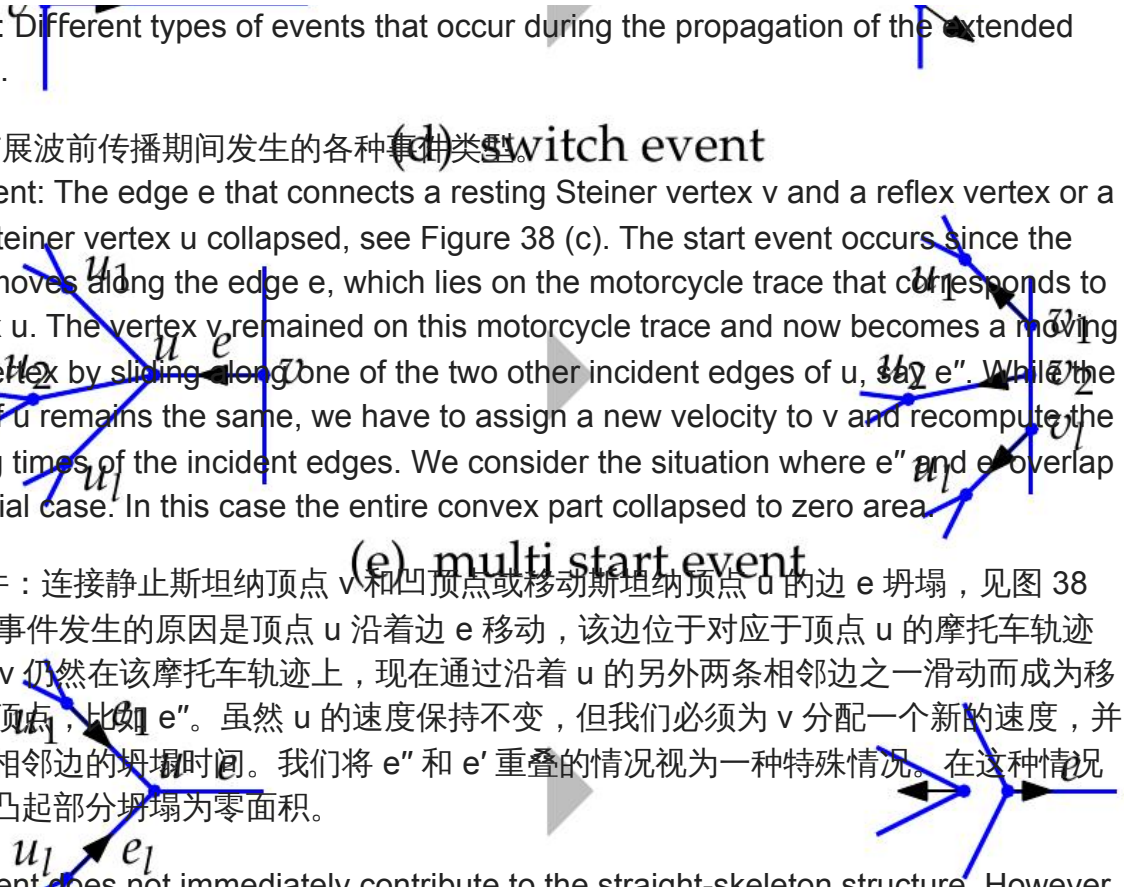


图 38：扩展波前传播期间发生的各种事件类型。

• Start event: The edge e that connects a resting Steiner vertex v and a reflex vertex or a moving Steiner vertex u collapsed, see Figure 38 (c). The start event occurs since the vertex u moves along the edge e, which lies on the motorcycle trace that corresponds to the vertex u. The vertex v remained on this motorcycle trace and now becomes a moving Steiner vertex by sliding along one of the two other incident edges of u, say e″. While the velocity of u remains the same, we have to assign a new velocity to v and recompute the collapsing times of the incident edges. We consider the situation where e″ and e overlap as a special case. In this case the entire convex part collapsed to zero area.

• 开始事件：连接静止斯坦纳顶点 v 和凹顶点或移动斯坦纳顶点 u 的边 e 坍塌，见图 38 (c)。开始事件发生的原因是顶点 u 沿着边 e 移动，该边位于对应于顶点 u 的摩托车轨迹上。顶点 v 仍然在该摩托车轨迹上，现在通过沿着 u 的另外两条相邻边之一滑动而成为移动斯坦纳顶点，比如 e″。虽然 u 的速度保持不变，但我们必须为 v 分配一个新的速度，并重新计算相邻边的坍塌时间。我们将 e″ 和 e′ 重叠的情况视为一种特殊情况。在这种情况下，整个凸起部分坍塌为零面积。

A start event does not immediately contribute to the straight-skeleton structure. However, altering the O(1) entries in Q costs us O(log n) time.

起始事件不会立即对直骨骼结构产生贡献。然而，更改Q中的O(1)个条目会花费我们O(log n)的时间。

• Switch event: The edge e between a convex vertex u and a reflex vertex or a moving Steiner vertex v collapsed, see Figure 38 (d). That means that the vertex u migrates from one convex face of the extended wavefront to a neighboring one, by jumping over the vertex v. As a consequence, the vertex v now splits the opposite edge incident to u. While the vertex u maintains its velocity, we need to recompute the velocity for the vertex v.

• 开关事件：凸顶点 u 和凹顶点或移动的斯坦纳顶点 v 之间的边 e 塌陷，参见图 38 (d)。这意味着顶点 u 从扩展波前的某个凸面迁移到相邻的一个凸面，跳过顶点 v。因此，顶点 v 现在分割与 u 相邻的相对边。当顶点 u 保持其速度时，我们需要重新计算顶点 v 的速度。

Similar to start events, a switch event does not directly contribute to the straight skeleton. Again it requires O(log n) time to update the corresponding entries in the priority queue Q.

与起始事件类似，切换事件不会直接影响直线骨架。同样，更新优先级队列Q中相应的条目需要O(log n)时间。

• Multi start event: The edge e that connects a moving Steiner vertex v and a multi Steiner vertex u collapsed, see Figure 38 (e). That means that the multi split event that corresponds to the multi Steiner vertex u will not happen, because the wavefront swept over the vertex u before.

• 多重启动事件：连接一个移动斯坦纳顶点v和一个多重斯坦纳顶点u的边e坍缩，见图38 (e)。这意味着对应于多重斯坦纳顶点u的多重分裂事件不会发生，因为波前在顶点u之前扫过。
Let us denote by u1, . . . , ul the other vertices that are incident to u. Assume that the edges e1, . . . , el, which connect u with u1, . . . , ul, respectively, appear in counterclockwise order at u such that the edge e is between e1 and el in the mentioned order. We remove the vertices u and v and introduce new moving Steiner vertices v1, . . . , vl such that vi moves towards ui, with 1 ≤ i ≤ l. The new vertices v1, . . . , vl are aligned on the supporting line of the two other edges that are incident to v. We add new edges uvi for 1 ≤ i ≤ l and vivi+1 for 1 ≤ i ≤ l − 1. Besides u, we have two additional vertices that were adjacent to v. We connect these two vertices to v1 and vl, respectively. The multi start event does not directly contribute to the straight-skeleton structure. Computing the collapsing times of the new edges and adapting the collapsing times of old edges requires O(l log n) time.

令 u1, . . . , ul 表示与 u 相邻的其他顶点。假设连接 u 与 u1, . . . , ul 的边 e1, . . . , el 在 u 处以逆时针顺序出现，使得边 e 在上述顺序中位于 e1 和 el 之间。我们移除顶点 u 和 v，并引入新的移动斯坦纳顶点 v1, . . . , vl，使得 vi 向 ui 移动，其中 1 ≤ i ≤ l。新的顶点 v1, . . . , vl 对齐于与 v 相邻的另外两条边的支撑线上。对于 1 ≤ i ≤ l，我们添加新的边 uvi，对于 1 ≤ i ≤ l − 1，我们添加新的边 vivi+1。除了 u 之外，我们还有两个与 v 相邻的额外顶点。我们将这两个顶点分别连接到 v1 和 vl。多起点事件不会直接影响直线骨架结构。计算新边的塌陷时间并调整旧边的塌陷时间需要 O(l log n) 时间。

• Multi split event: The edges e1, . . . , el that connect the reflex vertices v1, . . . , vl with the multi Steiner vertex u collapsed at the same time, see Figure 38 (f). We assume that e1, . . . , el appear in counter-clockwise order at u. The multi Steiner vertex u exists, since two or more motorcycles simultaneously crashed at this location. Note that it is possible that a new motorcycle was emanated from this location and hence there could be an additional edge e incident to u. This edge e lies on the motorcycle trace of the emanated motorcycle, see Figure 31. If this is the case we assume that e is between e1 and el. Also note that in this case the angle from el to e1 is reflex.

• 多重分裂事件：连接凹顶点 v1, . . . , vl 与多重斯坦纳顶点 u 的边 e1, . . . , el 同时坍缩，见图 38 (f)。我们假设 e1, . . . , el 以逆时针顺序出现在 u 处。多重斯坦纳顶点 u 存在，因为两个或多个摩托车同时在该位置碰撞。请注意，有可能从该位置发出新的摩托车，因此可能存在与 u 相邻的附加边 e。这条边 e 位于发出的摩托车的摩托车轨迹上，见图 31。如果是这种情况，我们假设 e 位于 e1 和 el 之间。另请注意，在这种情况下，从 el 到 e1 的角度是凹的。
Each consecutive pair uiui+1, with 1 ≤ i ≤ l − 1, gives rise to a new convex vertex similar to an ordinary split event. This new convex vertex is incident to two edges that were formerly the right arm of ui and the left arm of ui+1, respectively. Similar to the ordinary split event, we check whether the two incident edges are overlapping. In this case, the whole convex face of the extended wavefront collapsed as the multi split event happened.

每个连续对uiui+1，其中1 ≤ i ≤ l − 1，都会产生一个新的凸顶点，类似于一个普通的分割事件。这个新的凸顶点与两条边相关联，这两条边分别是ui的右臂和ui+1的左臂。与普通的分割事件类似，我们检查两条相关联的边是否重叠。在这种情况下，当多重分割事件发生时，扩展波前的整个凸面都会坍塌。
If the angle from el to e1 is also convex then there is no additional edge e incident to u and we proceed as above. Otherwise we have an additional edge e incident to u. If the right arm of ul and the left arm of e1 span a convex angle than the edge e and the old edge el are collinear, cf. Figure 31 (d). We remove the edge e and proceed as above. Otherwise, the right arm of ul and the left arm of e1 span a reflex angle. Then the edge e lies on their bisector and we introduce a new reflex vertex that drives on the edge e, cf. Figure 31 (c).

如果从el到e1的角度也是凸的，那么没有额外的边e与u相关联，我们像上面一样继续。否则，我们有另一条与u相关的边e。如果ul的右臂和e1的左臂构成一个凸角，那么边e和旧边el是共线的，参见图31（d）。我们移除边e，并如上所述继续。否则，ul的右臂和e1的左臂构成一个凹角。那么边e位于它们的角平分线上，我们引入一个新的凹顶点，它在边e上移动，参见图31（c）。

Each ui, with 1 ≤ i ≤ l, traced out a reflex straight-skeleton arc, which we add to our straight-skeleton structure. Computing the collapsing times of the new edges and altering the corresponding entries in Q takes O(l log n) time.

每个ui，其中1 ≤ i ≤ l，描绘出一个反射的直骨架弧，我们将其添加到我们的直骨架结构中。计算新边的坍塌时间并更改Q中相应的条目需要O(l log n)时间。

After the last event occurred, the extended wavefront has the shape of a polygon circumscribing the graph G. From each vertex v of this polygon that has a reflex angle on the outer side, there is an incident edge which reaches to infinity and which corresponds to a motorcycle that escaped. We add these infinite edges as reflex arcs to our straight skeleton structure and connect the infinite end nodes in a circular manner. This allows us to assume that the boundary of any straight-skeleton face f (e), including the unbounded faces, connects one endpoint of e with the other endpoint of e. We also refer to the discussion on unbounded faces and infinite arcs in Section 1.2.2.

在最后一个事件发生后，扩展波前具有一个外接图G的多边形的形状。从这个多边形的每个顶点v（其外侧有一个反射角）出发，都有一条延伸至无穷远的入射边，对应于一辆逃逸的摩托车。我们将这些无限边作为反射弧添加到我们的直线骨架结构中，并以环形方式连接无限端节点。这允许我们假设任何直线骨架面f(e)的边界，包括无界面，将e的一个端点与e的另一个端点连接起来。我们还参考了第1.2.2节中关于无界面和无限弧的讨论。

2.5.2 Runtime analysis

2.5.2 运行时分析

Let us assume that the motorcycle graph M(G) of G is already given. In order to create the extended wavefront W ∗(G, 0) at time zero we first create W (G, 0): We start with an edge e and a vertex v incident to e. We walk around the corresponding connected

component by setting e to the next counter-clockwise edge of e at v and by setting v to the opposite vertex of the updated e. We continue this procedure until we again arrive at our starting edge. At each step, we duplicate the current edge e and consider its duplication as the wavefront edge at one side of e. At any time when we wrap around at a terminal vertex, we add an additional wavefront edge. After we arrive again at the starting edge, we repeat the whole procedure for any edge of G that has not yet a duplicate on either side. This allows us to create W (G, 0) in O(n log n) time for an n-vertex graph G. Next, we insert M(G) into W (G, 0) in order to obtain W ∗(G, 0). Hence, every motorcycle that crashed into a wall — that is, into an edge e of G — splits the corresponding wavefront duplicate of the input edge e. We need to take into account that multiple motorcycles may crash into the same side of an input edge e. We sort the motorcycles along e and split the corresponding wavefront edge accordingly. To sum up, we can construct W ∗(G, 0) in O(n log n) time if M(G) is known.

我们假设 G 的摩托车图 M(G) 已经给出。为了在零时刻创建扩展波前 W ∗(G, 0)，我们首先创建 W (G, 0)：我们从一条边 e 和一个与 e 相邻的顶点 v 开始。我们通过将 e 设置为 e 在 v 处的下一个逆时针方向的边，并将 v 设置为更新后的 e 的相对顶点，来围绕相应的连通分量行走。我们继续这个过程，直到再次到达我们的起始边。在每个步骤中，我们复制当前边 e，并将其复制视为 e 一侧的波前边。在任何时候，当我们围绕一个终端顶点旋转时，我们都会添加一个额外的波前边。在我们再次到达起始边之后，我们对 G 的任何尚未在任一侧有副本的边重复整个过程。这允许我们在 O(n log n) 时间内为 n 顶点图 G 创建 W (G, 0)。接下来，我们将 M(G) 插入到 W (G, 0) 中，以获得 W ∗(G, 0)。因此，每辆撞到墙壁的摩托车——也就是撞到 G 的边 e——都会分割输入边 e 的相应波前副本。我们需要考虑到多辆摩托车可能会撞到输入边 e 的同一侧。我们沿着 e 对摩托车进行排序，并相应地分割相应的波前边。总而言之，如果 M(G) 已知，我们可以在 O(n log n) 时间内构造 W ∗(G, 0)。
The number of edge events is in O(n) and the number of split and start events is in O(r) ⊂ O(n), where r denotes the number of reflex wavefront vertices in the initial wavefront W (G, 0). However, as discussed in Section 2.3, the number k of switch events is in O(nr) ⊂ O(n2). Each multi start event and multi split event is handled in O(l log n) time, where the number l corresponds to the number of motorcycles that gave rise to this event. The sum of all values l among the multi start and multi split events is therefore in O(r) ⊂ O(n).

边事件的数量为O(n)，分裂和开始事件的数量为O(r) ⊂ O(n)，其中r表示初始波前W(G, 0)中的反射波前顶点的数量。然而，正如第2.3节所讨论的，切换事件的数量k为O(nr) ⊂ O(n2)。每个多重开始事件和多重分裂事件都在O(l log n)时间内处理，其中数量l对应于引起此事件的摩托车的数量。因此，多重开始和多重分裂事件中所有值l的总和为O(r) ⊂ O(n)。

Lemma 2.32. Let G denote a planar straight-line graph G with n vertices, where r denotes the number of reflex wavefront vertices in the initial wavefront W (G, 0). If M(G) is known then our algorithms runs in O((n + k) log n) time and O(n) space, where k ∈ O(nr) ⊂ O(n2) denotes the number of switch events occurred.

引理 2.32. 设 G 表示具有 n 个顶点的平面直线图 G，其中 r 表示初始波前 W(G, 0) 中反射波前顶点的数量。如果已知 M(G)，则我们的算法在 O((n + k) log n) 时间和 O(n) 空间内运行，其中 k ∈ O(nr) ⊂ O(n2) 表示发生的切换事件的数量。
As already mentioned in Section 2.3.3, the O(n2) bound for the number of switch events is tight. However, we want to recall that it appears to be very unlikely that more than Ω(n) switch events actually occur for practical applications and hence an O(n log n) runtime can be expected in real world. In Section 2.5.4 we present extensive experimental results on 13 500 datasets of different characteristics that demonstrate an O(n log n) runtime on virtually all of our datasets.

正如第2.3.3节中已经提到的，开关事件数量的O(n2)界限是紧的。然而，我们想回顾一下，对于实际应用来说，出现超过Ω(n)个开关事件的可能性似乎非常小，因此在现实世界中可以预期O(n log n)的运行时间。在第2.5.4节中，我们展示了对13 500个具有不同特征的数据集进行的大量实验结果，这些结果表明几乎在我们所有的数据集上都具有O(n log n)的运行时间。

In order to compute the motorcycle graph M(G), we need an algorithm that supports the dynamic insertion of motorcycles during the simulation. Hence, we cannot apply the algorithm by Cheng and Vigneron, cf. Section 1.4.2.4, which needs to compute the 1/√r-cutting a-priori. However, the algorithm by Eppstein and Erickson [EE99], cf. Section 1.4.2.3, employs dynamic data structures which are capable of adding new motorcycles during the simulation. Hence, in theory, we can compute the motorcycle graph M(G) in O(r17/11+ϵ) time and space. In Section 3.3 we will discuss a practical algorithm that runs in O(r log r) time on average, under the assumption that the motorcycles are distributed uniformly enough.

为了计算摩托车图M(G)，我们需要一种算法，该算法支持在模拟过程中动态插入摩托车。因此，我们不能应用Cheng和Vigneron的算法，参见第1.4.2.4节，该算法需要先验地计算1/√r-cutting。然而，Eppstein和Erickson [EE99]的算法，参见第1.4.2.3节，采用了动态数据结构，能够在模拟过程中添加新的摩托车。因此，理论上，我们可以在O(r17/11+ϵ)时间和空间内计算摩托车图M(G)。在第3.3节中，我们将讨论一种实际的算法，在摩托车分布足够均匀的假设下，该算法平均在O(r log r)时间内运行。

2.5.3 Details of the implementation Bone

2.5.3 骨骼实现的细节

We casted our algorithm into the implementation Bone. Bone is written in C++ and uses ordinary double-precision floating-point arithmetic according to IEEE-754. For standard data structures, such as stacks, queues, maps (red-black trees), etc., we use the Standard Template Library. The motorcycle graph M(G) is computed by our motorcycle graph code Moca, see Section 3.3. One central component of Bone is a kinetic straight-line graph that assigns to each vertex a speed ray (the position at time zero and a velocity). Furthermore, each non-isolated vertex has an index to an incident edge and each edge has, for both endpoints, references to the next clockwise and counter-clockwise edges. The kinetic graph models the extended wavefront and does not

maintain planarity. It is in the responsibility of the event handling code to maintain the proper topology of the kinetic graph. However, this graph structure allows us to easily remove edges and reconnect vertices, which is frequently done in the event handling procedures.

我们将算法转化为 Bone 的实现。Bone 是用 C++ 编写的，并根据 IEEE-754 标准使用普通的双精度浮点算术。对于标准数据结构，如栈、队列、映射（红黑树）等，我们使用标准模板库。摩托车图 M(G) 由我们的摩托车图代码 Moca 计算，参见第 3.3 节。Bone 的一个核心组件是一个动力学直线图，它为每个顶点分配一个速度射线（零时刻的位置和速度）。此外，每个非孤立顶点都有一个指向入射边的索引，并且每条边对于两个端点都有指向下一个顺时针和逆时针边的引用。动力学图对扩展波前进行建模，但不保持平面性。事件处理代码负责维护动力学图的正确拓扑结构。然而，这种图结构使我们能够轻松地移除边和重新连接顶点，这在事件处理过程中经常进行。
For the actual implementation it turned out that is advantageous to introduce a new type of vertex, the so-called multi convex vertex. In a degenerate case, it could happen that a convex wavefront vertex and a moving Steiner vertex move along the same trajectory. For instance, consider a symmetric non-convex 4-gon as input, where the motorcycle within the 4-gon crashes exactly into the opposite convex vertex. In this case it is important, for practical reasons, to merge the convex vertex and the Steiner vertex into a multi convex vertex.

在实际实现中，引入一种新型顶点，即所谓的多凸顶点，是有利的。在退化情况下，可能会发生凸波前顶点和移动的斯坦纳顶点沿同一轨迹移动的情况。例如，考虑一个对称的非凸四边形作为输入，其中四边形内的摩托车恰好撞入相对的凸顶点。在这种情况下，出于实际原因，将凸顶点和斯坦纳顶点合并为多凸顶点非常重要。

2.5 the general wavefront-type algorithm 73

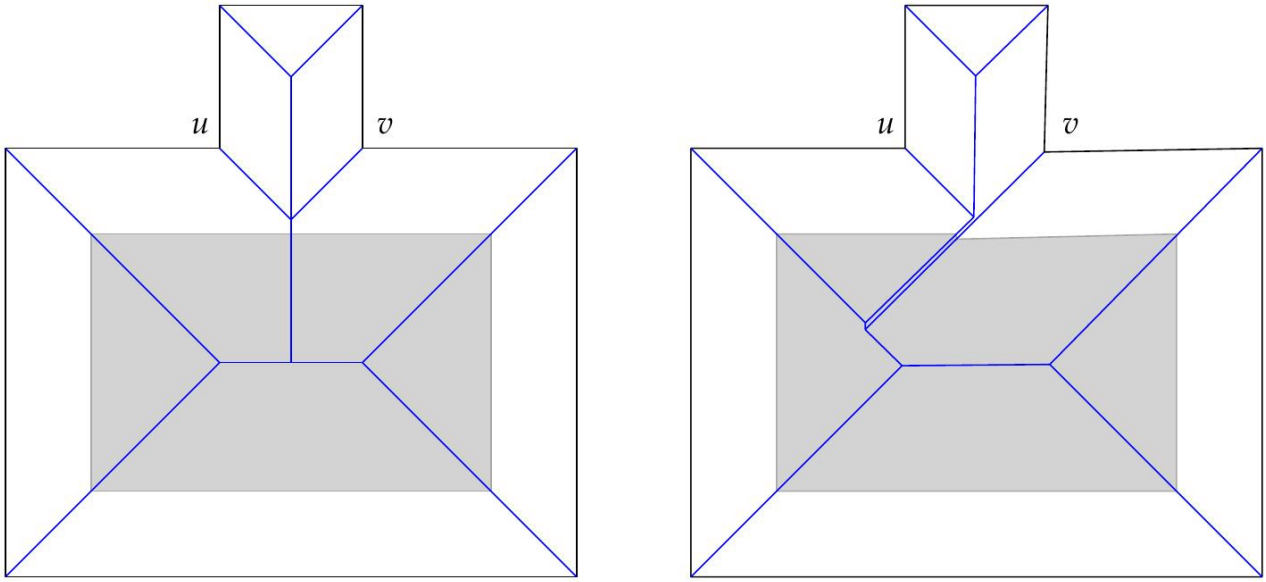


Figure 39: An arbitrarily small perturbation of a vertex v significantly changes the straight skeleton. Left: v is involved in a vertex event with the vertex u. Right: v has been slightly dislocated to the interior of the left polygon. Since v is slightly faster than u, the vertex event is avoided and the reflex arc incident to v is dramatically longer. The corresponding offset polygons are depicted in gray. Note that the offset polygon in the right subfigure contains a reflex vertex at the top.

图 39：顶点 v 的一个任意小的扰动会显著改变直线骨架。左图：v 参与了与顶点 u 的一个顶点事件。右图：v 已稍微移位到左侧多边形的内部。由于 v 比 u 稍微快一些，因此避免了顶点事件，并且与 v 相关的凹弧明显更长。相应的偏移多边形以灰色描绘。请注意，右侧子图中的偏移多边形在顶部包含一个凹顶点。

For real-world implementations of geometric algorithms, it is advisable to implement some a posteriori checks that test whether the result fulfills some basic but necessary properties in order to be correct. Bone follows that strategy and provides a posteriori checks that test whether the boundary of the faces are connected.

对于几何算法的实际应用，建议实施一些后验检查，以测试结果是否满足一些基本但必要的属性，以确保其正确性。Bone 遵循该策略，并提供后验检查，以测试面的边界是否已连接。

The fact that the straight skeleton S (G) does not continuously depend on the input graph G — say, in terms of the Hausdorff distance — has important practical implications. First of all, we cannot apply perturbation techniques in order to resolve special cases in various algorithms, as already mentioned by Eppstein and Erickson [EE99]. Secondly, we have to be cautious when we consider the limit of a sequence of straight skeletons. For

instance, image we define the wavefront emanated by an isolated vertex by approximating the vertex by small squares or small segments. (Recall the discussion on the wavefront emanated from a isolated vertex in Section 1.2.2.) The limit of the straight skeletons does not necessarily result in the desired structure and may depend on the actual sequence of input graphs. Thirdly, the question arises how to determine whether two motorcycles or two reflex vertices meet each other at exactly the same time.

直线骨架S(G)不连续地依赖于输入图[NT0]G的事实——例如，就Hausdorff距离而言——具有重要的实际意义。首先，我们不能应用扰动技术来解决各种算法中的特殊情况，正如Eppstein和Erickson [EE99]已经提到的那样。其次，当我们考虑直线骨架序列的极限时，我们必须谨慎。例如，假设我们通过用小正方形或小线段逼近顶点来定义由孤立顶点发出的波前。（回顾第1.2.2节中关于从孤立顶点发出的波前的讨论。）直线骨架的极限不一定会产生所需的结构，并且可能取决于输入图的实际序列。第三，出现了一个问题，即如何确定两个摩托车或两个凹顶点是否在完全相同的时间相遇。

• If we use ordinary double-precision floating-point arithmetic then we need to apply ϵ-based comparisons. That is, we define a small positive constant ϵ and in order to test whether the numbers a and b are equal we tests whether $|a - b| \leq \epsilon$.

• 如果我们使用普通的双精度浮点算术，那么我们需要应用基于 ε 的比较。也就是说，我们定义一个小的正数常量 ϵ，为了测试数字 a 和 b 是否相等，我们测试 $|a - b| \leq \varepsilon$ 是否成立。

• In order to avoid well-known problems introduced by ϵ-based comparisons, one could apply so-called exact predicates. The decision whether two values a and b are equal is based on the expression trees for the values a and b and the concept of so-called root-bounds [Yap04]. That is, one can precisely decide whether $|a - b|$ is zero, by evaluating the term $|a - b|$ with finite but arbitrary precision. The root bounds tell how precisely the term needs to be evaluated.

• 为了避免基于 ε 比较所引入的众所周知的问题，可以应用所谓的精确谓词。判断两个值 a 和 b 是否相等，是基于值 a 和 b 的表达式树以及所谓的根界限 [Yap04] 的概念。也就是说，可以通过以有限但任意的精度评估项 $|a - b|$，来精确地判断 $|a - b|$ 是否为零。根界限说明了需要以多高的精度来评估该项。

Let us consider the application of roof construction, cf. Section 1.2.3. Assume that we implement a roof-construction software based on straight skeletons, which reads the wall footprint of a house from an input file. Since the straight skeleton is sensitive to small errors on the input we would require that the input file is capable of carrying exact coordinates. This basically means that we need to save expression trees of all coordinates in the input files. However, this seems to be infeasible for real-world applications:

让我们考虑屋顶建造的应用，参见第1.2.3节。假设我们实现了一个基于直线骨架的屋顶建造软件，它可以从输入文件中读取房屋的墙体轮廓。由于直线骨架对输入中的小误差非常敏感，我们需要输入文件能够携带精确的坐标。这基本上意味着我们需要在输入文件中保存所有坐标的表达式树。然而，对于实际应用来说，这似乎是不可行的：

• The file size would skyrocket and, moreover, the expression trees tend to grow in size as the input gets more and more complex, because the coordinates of new vertices tend to depend on the positions of previous vertices.

• 文件大小会急剧增加，而且，随着输入变得越来越复杂，表达式树的大小往往会增长，因为新顶点的坐标往往取决于先前顶点的位置。

• Standard file formats, like the DXF format, do not support the embedding of expression trees for the coordinates in a standardized way.

• 诸如DXF格式之类的标准文件格式，并不以标准化的方式支持坐标表达式树的嵌入。

• We would require that common CAD software uses exact arithmetic kernels for their own calculations (e. g., snapping to intersection points). In the end, we would require that the entire work-flow uses exact arithmetic.

• 我们要求常用的CAD软件使用精确算术内核进行自身计算（例如，捕捉到交点）。最终，我们要求整个工作流程都使用精确算术。

However, if we assume that the coordinates in the input files are imprecise due to the limited precision or ordinary floating-point numbers, we are basically forced to resort to ϵ-based comparisons for real-world applications. Otherwise, it is virtually impossible to construct non-trivial input data, for which we intend that a vertex event happens. Assume we would like to construct a symmetric wall footprint as in Figure 39 (a), which would produce a roof, where two valleys meet in a common endpoint. However, due to numerical imprecisions, we instead obtain a roof that is based on Figure 39 (b).

然而，如果我们假设输入文件中的坐标由于精度有限或普通浮点数而不精确，那么我们基本上不得不求助于基于 ε 的比较，以用于实际应用。否则，几乎不可能构建非平凡的输入数据，而我们希望顶点事件发生。假设我们想要构建一个如图 39 (a) 所示的对称墙体轮廓，这将产生一个屋顶，其中两个谷在一个共同的端点相遇。然而，由于数值上的不精确性，我们得到的屋顶是基于图 39 (b) 的。

Note that for Voronoi diagrams the problem of imprecise input is not as severe as for straight skeletons, because the error introduced to the Voronoi diagrams is, roughly speaking, of the same magnitude as the error embedded in the input values.

请注意，对于Voronoi图，不精确输入的问题不像对于直线骨架那样严重，因为引入到Voronoi图的误差，粗略地说，与嵌入在输入值中的误差大小相同。

2.5.4 Experimental results and runtime statistics

2.5.4 实验结果与运行时统计

In this subsection we present experimental results for our implementation Bone and the straight-skeleton implementation that is shipped with CGAL 3.8, see Section 1.4.3. As already mentioned, Bone is able to process planar straight-line graphs as input, but the

implementation in CGAL can only process polygons with holes. This is the reason why we restrict the following runtime statistics to datasets that contain polygons with holes only.

在本小节中，我们将展示我们的实现 Bone 和 CGAL 3.8 附带的直线骨架实现的实验结果，参见第 1.4.3 节。正如已经提到的，Bone 能够处理平面直线图作为输入，但 CGAL 中的实现只能处理带孔的多边形。这就是为什么我们将以下运行时统计数据限制为仅包含带孔多边形的数据集的原因。
Since Bone computes the straight skeleton inside and outside the input polygon we call the straight-skeleton routines of CGAL outside the polygon, inside the polygon and also inside the holes, if any exist.[2]

由于 Bone 计算输入多边形内部和外部的直线骨架，因此我们调用 CGAL 的直线骨架例程，分别在多边形外部、多边形内部以及孔内部（如果存在）进行计算。[2]

We ran our runtime tests on a about 13 500 datasets, comprising realistic and contrived data of different characteristics, including mechanical designs, printed circuit boards, font outlines, random polygons generated by RPG [AH96], space filling curves, and fractal datasets. The size of the datasets ranges from only a few dozen vertices up to several millions.

我们对大约 13 500 个数据集进行了运行时测试，这些数据集包含具有不同特征的真实和人为数据，包括机械设计、印刷电路板、字体轮廓、RPG [AH96] 生成的随机多边形、空间填充曲线和分形数据集。数据集的大小从几十个顶点到几百万个不等。

Our experiments were carried out on a Linux machine with a 64-bit kernel, running on an Intel Core i7 processor clocked at 3.33 GHz. Note that this processor provides several cores, but neither Bone nor the straight-skeleton implementation from CGAL is multi-threaded. We avoided a falsified system behavior — e. g., caused by the invocation of the out-ofmemory killer of Linux, freeing of all file system buffers, etc. — by restricting the memory utilization of Bone resp. the CGAL code to 6 GiB, using the ulimit command. Furthermore, we restricted the runtime for a single dataset to 10 minutes. The time consumption in order to compute the straight skeleton was measured by considering the user-space time consumption retrieved by the C-function getrusage.

我们的实验在一台具有 64 位内核的 Linux 机器上进行，该机器运行在主频为 3.33 GHz 的 Intel Core i7 处理器上。请注意，该处理器提供多个内核，但 Bone 和 CGAL 中的直线骨架实现都不是多线程的。我们通过限制 Bone 的内存使用量，避免了伪造的系统行为——例如，由 Linux 的内存不足杀手程序的调用、所有文件系统缓冲区的释放等引起——。CGAL 代码限制为 6 GiB，使用 ulimit 命令。此外，我们将单个数据集的运行时间限制为 10 分钟。为了计算直线骨架的时间消耗，我们考虑了通过 C 函数 getrusage 检索到的用户空间时间消耗。

Figure 40 shows four plots with identical axes and in each plot a blue dot represents the runtime of a single dataset. Note that both axes use a logarithmic scale. The x-axis shows the size n of the dataset and the y-axis illustrates the runtime for a dataset in seconds.

For illustrative reasons we divided the runtime for each dataset of size n by the factor n log n. Hence, a horizontal line in the plot corresponds to a runtime complexity of n log n and a line with slope 1 corresponds to a runtime complexity of n2 log n.

图40显示了四个具有相同坐标轴的图，每个图中一个蓝点代表一个数据集的运行时间。请注意，两个坐标轴都使用了对数刻度。x轴显示数据集的大小n，y轴显示数据集的运行时间（以秒为单位）。出于说明目的，我们将大小为n的每个数据集的运行时间除以因子n log n。因此，图中一条水平线对应于n log n的运行时间复杂度，一条斜率为1的线对应于n2 log n的运行时间复杂度。
In subfigure (a) we plotted the runtime statistics for Bone. As predicted, Bone exhibits an n log n runtime behavior on the vast majority of our datasets. All dots in the gray area correspond to an actual runtime of 10 to 30 · n log n µs, where n varies from 60 up to $2 \cdot 10^6$. The motorcycle graph M(G) is computed by Moca, see Section 3.3. This implementation exhibits an O(n log n) runtime for well distributed input. We extensively investigate the runtime of Moca in Section 3.3.3. Our theoretical runtime analysis in Section 2.5.2 at first focuses on the assumption that the motorcycle graph is already available. For this reason, we plotted the runtime consumption of Bone, without the time consumed for the computation of the motorcycle graph M(G), in subfigure (b). We observe that the few outliers in subfigure (a) were virtually all caused by the computation of the motorcycle graph. Hence, an actual runtime of O(n log n) holds for basically all datasets from our database.

在子图 (a) 中，我们绘制了 Bone 的运行时统计数据。正如预测的那样，Bone 在我们绝大多数数据集上表现出 n log n 的运行时行为。灰色区域中的所有点都对应于 10 到 30 · n log n µs 的实际运行时，其中 n 的变化范围从 60 到 $2 \cdot 10^6$。摩托车图 M(G) 由 Moca 计算，参见第 3.3 节。对于分布良好的输入，此实现表现出 O(n log n) 的运行时。我们在第 3.3.3 节中广泛研究了 Moca 的运行时。我们在第 2.5.2 节中的理论运行时分析首先侧重于摩托车图已经可用的假设。因此，我们在子图 (b) 中绘制了 Bone 的运行时消耗，不包括计算摩托车图 M(G) 所消耗的时间。我们观察到子图 (a) 中的少数异常值几乎都是由摩托车图的计算引起的。因此，O(n log n) 的实际运行时基本上适用于我们数据库中的所有数据集。
In subfigure (c) we plotted the runtime of CGAL using an arithmetic kernel with exact predicates but inexact constructions. By default, the CORE [Cor] library is employed as the arithmetic backend. Using exact predicates should guarantee that the topology of the straight skeleton is correct, even if the locations of the nodes are not necessarily exact.[3] However, using exact constructions makes the runtime consumption to skyrocket. That is, datasets with a few hundred vertices lead to runtimes in the range of minutes. We see that the runtime consumption of CGAL is mostly in the gray area that represents runtimes between 0.17 to 1.7 · n2 log n µs. In particular, CGAL exhibits at least a quadratic runtime consumption in practice. We also observe that the runtimes of different datasets of the same size vary within two decades. While many datasets with about a hundred vertices need roughly 20 ms, many other datasets lead to runtimes of roughly 20 seconds.

在子图 (c) 中，我们绘制了 CGAL 的运行时间，它使用了一个具有精确谓词但非精确构造的算术内核。默认情况下，CORE [Cor] 库被用作算术后端。使用精确谓词应保证直线骨架的拓扑结构是正确的，即使节点的位置不一定是精确的。[3] 然而，使用精确构造会使运行时

间消耗急剧增加。也就是说，具有几百个顶点的数据集会导致运行时间达到几分钟的范围。我们看到 CGAL 的运行时间消耗主要在灰色区域，该区域表示 0.17 到 1.7 · n2 log n µs 之间的运行时间。特别地，CGAL 在实践中至少表现出二次方的运行时间消耗。我们还观察到，相同大小的不同数据集的运行时间在两个数量级内变化。虽然许多具有大约一百个顶点的数据集大约需要 20 毫秒，但许多其他数据集会导致大约 20 秒的运行时间。

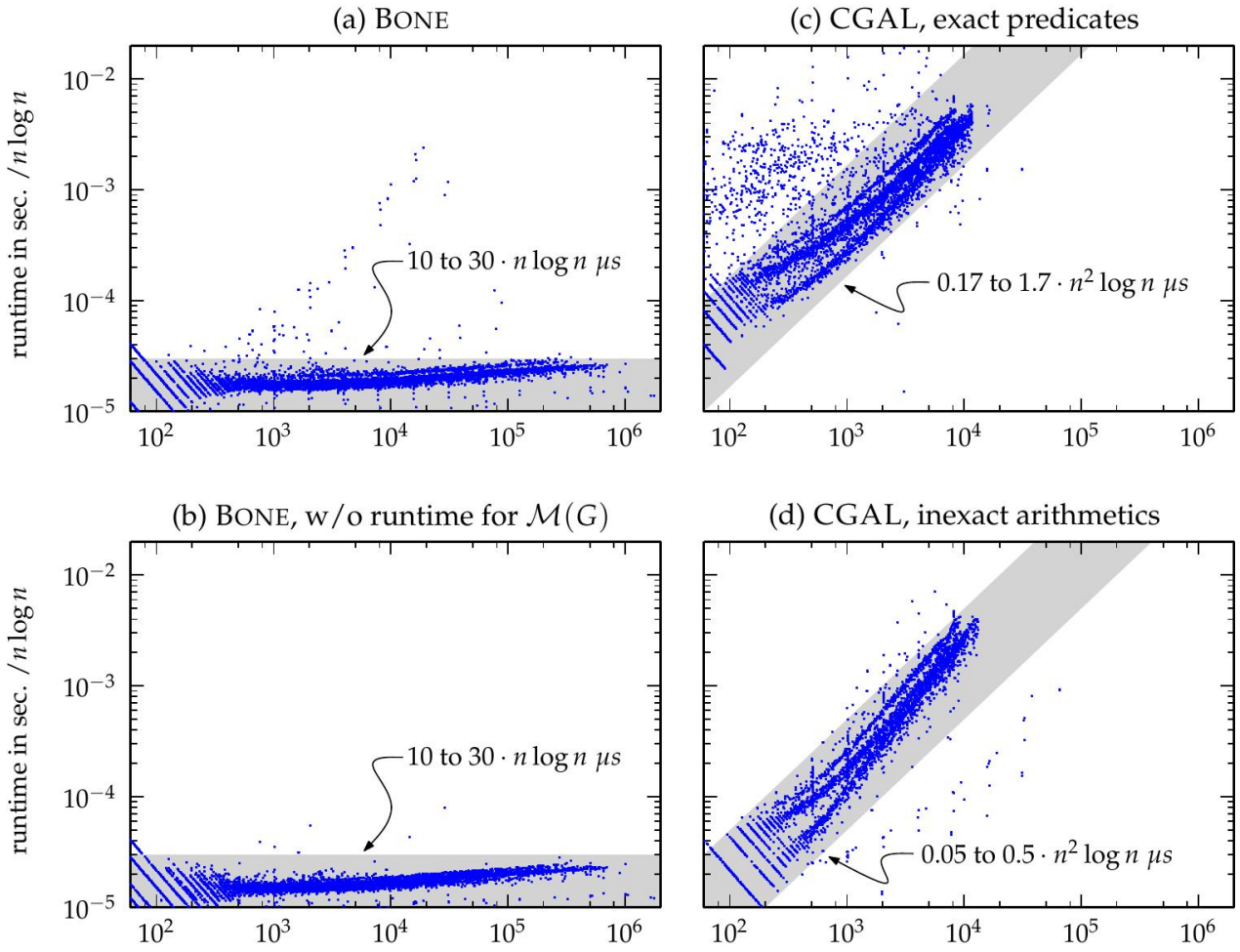


Figure 40: Every point shows the runtime for a single dataset. The x-axis denotes the number n of vertices of the input, ranging from 60 to 2 · 10^6. For illustrative reasons we divided the actual runtime in seconds by n log n. Hence, a horizontal line corresponds to an n log n complexity. Points in the shaded areas correspond to runtimes as labeled.

图 40：每个点表示单个数据集的运行时间。x 轴表示输入的顶点数 n，范围从 60 到 2 · 10^6。为了便于说明，我们将实际运行时间（以秒为单位）除以 n log n。因此，水平线对应于 n log n 的复杂度。阴影区域中的点对应于标记的运行时间。

In subfigure (d) we plotted the runtime of CGAL using inexact arithmetics in order to (i) gain a better insight on the runtime penalty due to the exact predicates kernel and (ii) to obtain a better comparison to Bone. We first note that CGAL, with inexact arithmetics, still exhibits at least a quadratic runtime complexity. The vast majority of datasets is processed in 0.05 to 0.5 · n2 log n µs. However, the big deviations for the runtimes among datasets of the same size vanished. We also observe that using inexact

arithmetics increases the performance by a factor of about 3 to 100. However, we want to note that the straight-skeleton implementation of CGAL is not supposed to be used with an inexact predicates kernel.[4]

在子图 (d) 中，我们绘制了 CGAL 使用非精确算术的运行时间，以便 (i) 更好地了解由于精确谓词内核造成的运行时间损失，以及 (ii) 获得与 Bone 更好的比较。我们首先注意到，即使使用非精确算术，CGAL 仍然表现出至少是二次方的运行时间复杂度。绝大多数数据集的处理时间在 0.05 到 0.5 · n2 log n µs 之间。然而，相同大小的数据集之间运行时间的巨大偏差消失了。我们还观察到，使用非精确算术可以将性能提高大约 3 到 100 倍。但是，我们想指出的是，CGAL 的直骨架实现不应该与非精确谓词内核一起使用。[4]
memory usage Bone was able to handle input containing more than 10^6 vertices. However, CGAL could only handle datasets with up to about 10 000 vertices within the time and space constraints mentioned above. Interestingly, also CGAL with inexact arithmetics was not able to compute datasets containing significantly more than about 10 000 vertices within the 6 GiB memory limit. We listed the memory footprint of Bone and CGAL on random datasets that were generated by RPG in Table 3. The memory usage was measured by the library libmemusage.so, which is shipped with glibc [GLI]. This library hooks into the malloc calls of a process and accounts the memory usage. In Table 3, we print the peak heap size in megabytes and the relative change to the previous row. Note that the actual memory footprint of a process is larger due to ordinary memory fragmentation in the heap and, of course, the program code, libraries and the stack require a few megabytes as well. For instance, the CGAL process required in total almost 20 GiB virtual memory in order to process the last dataset with 16 384 vertices.

内存使用方面，Bone能够处理包含超过10^6个顶点的输入。然而，CGAL在上述时间和空间约束内，只能处理最多约10 000个顶点的数据集。有趣的是，即使使用非精确算法的CGAL，也无法在6 GiB内存限制内计算包含明显多于约10 000个顶点的数据集。我们在表3中列出了Bone和CGAL在RPG生成的随机数据集上的内存占用。内存使用情况由库libmemusage.so测量，该库随glibc [GLI]一起发布。该库挂钩到进程的malloc调用并计算内存使用情况。在表3中，我们打印了以兆字节为单位的峰值堆大小以及相对于前一行的相对变化。请注意，由于堆中常见的内存碎片，以及程序代码、库和堆栈也需要一些兆字节，因此进程的实际内存占用量更大。例如，CGAL进程总共需要近20 GiB的虚拟内存才能处理具有16 384个顶点的最后一个数据集。

Size n	Bone		CGAL	
	MB	factor	MB	factor
256	1.44		3.77	
512	2.65	1.8x	13.4	3.5x
1 024	5.06	1.9x	51.1	3.8x
2 048	9.86	1.9x	201	3.9x
4 096	19.5	2.0x	792	3.9x
8 192	38.7	2.0x	3 197	4.0x
16 384	77.1	2.0x	12 600	3.9x

Table 3: The memory usage of Bone and CGAL on random datasets generated by RPG. Each row contains the peak heap size in megabytes and the factor by which the size changed w. r. t. the previous row.

表 3：Bone 和 CGAL 在 RPG 生成的随机数据集上的内存使用情况。每行包含以兆字节为单位的峰值堆大小，以及相对于前一行的尺寸变化因子。

We observe that Bone doubles its memory footprint if the input size of the dataset is doubled. In contrast, CGAL quadruples its memory footprint if the input size is doubled, which suggests a quadratic space complexity. As we learned in a personal mail correspondence with F. Cacciola, the author of the straight-skeleton code in CGAL, the algorithm computes potential split events for all pairs of reflex wavefront vertices and wavefront edges and inserts them into a priority queue. This explains the O(n2) memory footprint and the O(n2 log n) runtime.

我们观察到，如果数据集的输入大小翻倍，Bone 的内存占用也会翻倍。相比之下，如果输入大小翻倍，CGAL 的内存占用会变为原来的四倍，这表明其空间复杂度为二次方级别。正如我们从与 CGAL 中直线骨架代码的作者 F. Cacciola 的私人邮件通信中了解到的那样，该算法计算所有反射波前顶点和波前边的对的潜在分裂事件，并将它们插入到优先级队列中。这解释了 O(n2) 的内存占用和 O(n2 log n) 的运行时间。

We tested the reliability of Bone on additional datasets, which do not necessarily form polygons with holes. Bone is also able to sample circular arcs from input files by straightline segments. This allows us to test Bone on basically the same datasets as we did for Vroni [HH09a]. Furthermore, Bone is able to compute offset-curves based on the straight-skeleton, which turned out to be a versatile development tool in order to find errors in the straight skeleton by visual inspections. Furthermore, Bone can export the terrain T (G) into a standard file format that can be read by 3D modeling software, like

Blender [Ble]. A wrong straight skeleton or significant numerical errors cause visible artifacts like non-planar terrain faces or very odd-looking terrains. Bone also contains many sanity checks for the validity of the wavefront during its propagation simulation and Bone is able to perform simple a posteriori checks on the resulting straight skeleton as mentioned in Section 2.5.3. It turns out that Bone performs well on our datasets in general. However, in order to boost Bone to industrial-strength, we need to implement some fine-tuning in the presence of wavefrontparts that simultaneously collapse in a self-parallel manner. However, the concept of the extended wavefront allows us to handle this issue on each convex faces of the extended wavefront separately, which appears to be a significant advantage.

我们测试了 Bone 在额外数据集上的可靠性，这些数据集不一定形成带孔的多边形。Bone 还可以通过直线段从输入文件中采样圆弧。这使得我们基本上可以在与 Vroni [HH09a] 相同的数据集上测试 Bone。此外，Bone 能够计算基于直骨架的偏移曲线，这被证明是一种通用的开发工具，可以通过视觉检查来发现直骨架中的错误。此外，Bone 可以将地形 T (G) 导出为标准文件格式，该格式可以被 3D 建模软件读取，例如 Blender [Ble]。错误的直骨架或显著的数值误差会导致可见的伪影，例如非平面地形面或看起来非常奇怪的地形。Bone 还包含许多健全性检查，用于验证波前在其传播模拟期间的有效性，并且 Bone 能够对生成的直骨架执行简单的后验检查，如第 2.5.3 节所述。事实证明，Bone 在我们的数据集上总体表现良好。然而，为了将 Bone 提升到工业强度，我们需要在波前部分以自平行方式同时坍缩的情况下进行一些微调。然而，扩展波前的概念允许我们分别处理扩展波前的每个凸面上的这个问题，这似乎是一个显著的优势。

2.6 s u m m a r y

2.6 总结

At the beginning of this chapter, the most promising approach towards a straight-skeleton implementation, in terms of implementability and real-world performance, appeared to be the triangulation-based algorithm by Aichholzer and Aurenhammer [AA98]. However, from a theoretical point of view, the best known worst-case time complexity is O(n3 log n) for an n-vertex planar straight-line graph. In Section 2.2, we used this circumstance as an entry point to our investigations. After presenting new aspects regarding the number of flip events, we were able to show that Steiner triangulations can be used in order to completely avoid flip events. This insight motivated a new approach to a straight-skeleton algorithm for non-degenerate polygons which is based on motorcycle graphs in Section 2.3. However, in order to generalize this algorithm to arbitrary planar straight-line graphs we had to carefully extend the motorcycle graph such that essential properties, which are required for our algorithm, are preserved, see Section 2.4. This generalization of the motorcycle graph allowed us to extend the approach from Section 2.3 to arbitrary planar straight-line graphs in Section 2.5.

在本章的开头，就可实现性和实际性能而言，最有希望的直线骨架实现方法似乎是Aichholzer和Aurenhammer提出的基于三角剖分的算法[AA98]。然而，从理论角度来看，对于一个具有n个顶点的平面直线图，目前已知的最坏情况时间复杂度是O(n3 log n)。在2.2节中，我们将这种情况作为我们研究的切入点。在介绍了关于翻转事件数量的新方面之后，我们能够证明可以使用Steiner三角剖分来完全避免翻转事件。这一见解促使我们在2.3

节中提出了一种基于摩托车图的非退化多边形直线骨架算法的新方法。然而，为了将该算法推广到任意平面直线图，我们必须仔细扩展摩托车图，以保持算法所需的基本属性，见2.4节。摩托车图的这种推广使我们能够将2.3节中的方法推广到2.5节中的任意平面直线图。
The resulting implementation Bone is able to handle planar straight-line graphs as input and exhibits a runtime of O(n log n) in practice. Compared to the current state-of-the-art straight-skeleton code, which is shipped with CGAL, this constitutes an improvement of a linear factor in time and space resp. a speed-up of one to two orders of magnitude for medium-sized datasets. Furthermore, CGAL is, in contrast to Bone, only able to handle polygons with holes as input. Our experimental results in Section 2.5.4 also revealed that Bone was able to process datasets containing more than a million vertices, while the implementation in CGAL was not able to handle datasets with significantly more than 10 000 vertices within the limits of 6 GiB memory.

最终实现的 Bone 能够处理平面直线图作为输入，并且在实践中表现出 O(n log n) 的运行时间。与 CGAL 附带的当前最先进的直线骨架代码相比，这在时间和空间上分别构成了一个线性因子的改进。对于中等大小的数据集，速度提高了一个到两个数量级。此外，与 Bone 相比，CGAL 只能处理带孔的多边形作为输入。我们在第 2.5.4 节中的实验结果还表明，Bone 能够处理包含超过一百万个顶点的数据集，而 CGAL 中的实现在 6 GiB 内存限制内无法处理包含明显超过 10 000 个顶点的数据集。

The theoretical worst-case runtime complexity of Bone is O(n2 log n), which is an improvement of a linear factor compared to the algorithm of Aichholzer and Aurenhammer [AA98]. However, the circumstances for which the wort-case occurs are easy to investigate for the algorithm underneath Bone, see Section 2.5. In fact, it appears to be very unlikely that the worst case actually happens for practical input, which has been confirmed by our experimental results in Section 2.5.4. Bone computed the straight skeleton for virtually all of our 13 500 datasets in 10 to 30 · n log n µs for datasets containing n vertices. This makes Bone the first straight-skeleton implementation that has been shown to run in O(n log n) in practice and which accepts planar straight-line graphs as input.

Bone 的理论最坏情况运行时间复杂度为 O(n2 log n)，与 Aichholzer 和 Aurenhammer [AA98] 的算法相比，这是一个线性因子的改进。然而，对于 Bone 底层算法来说，最坏情况发生的条件很容易研究，见第 2.5 节。事实上，最坏情况实际发生在实际输入中的可能性似乎非常小，这已通过我们在第 2.5.4 节中的实验结果得到证实。对于包含 n 个顶点的数据集，Bone 在 10 到 30 · n log n µs 内计算了几乎所有 13 500 个数据集的直线骨架。这使得 Bone 成为第一个被证明在实践中以 O(n log n) 运行，并且接受平面直线图作为输入的直线骨架实现。
Motorcycle graphs are strongly related to straight skeletons for several reasons. First of all, the motorcycle graph extracts the essential sub problem of computing straight skeletons. Secondly, the motorcycle graph and the straight skeleton possess a strong geometric relation, which is expressed by Theorem 2.11 and Theorem 2.26. Thirdly, motorcycle graphs help us to devise algorithms and implementations for the computation of straight skeletons. At the moment, the fastest algorithm for the straight skeleton of non-

degenerate polygons due to Cheng and Vigneron [CV07], see Section 1.4.2.4, and the fastest implementation, Bone, both employ the motorcycle graph. Hence, the investigation of motorcycle graphs is vital in order to obtain fast straight-skeleton algorithms and implementations and it is important to explore the motorcycle graph in order to get a better understanding of straight skeletons.

摩托车图与直线骨架密切相关，原因有以下几点。首先，摩托车图提取了计算直线骨架的本质子问题。其次，摩托车图和直线骨架具有很强的几何关系，这由定理2.11和定理2.26所表达。第三，摩托车图帮助我们设计用于计算直线骨架的算法和实现。目前，Cheng和Vigneron [CV07]提出的用于计算非退化多边形的直线骨架的最快算法（参见第1.4.2.4节）以及最快的实现Bone都采用了摩托车图。因此，为了获得快速的直线骨架算法和实现，对摩托车图的研究至关重要，并且为了更好地理解直线骨架，探索摩托车图非常重要。
In this chapter, we develop an implementation for the computation of the generalized motorcycle graph that works fast in practice. We start with a stochastic consideration of the trace lengths in a motorcycle graph in Section 3.2. It turns out that if n motorcycles are distributed well in the unit square then the average trace length is about $O(1/\sqrt{n})$. This insight motivates a motorcycle graph implementation that is based on geometric hashing in Section 3.3. Runtime experiments show that our implementation Moca exhibits an O(n log n) runtime for most of the 22 000 datasets tested. In Section 3.4, we further investigate the geometric relation between the motorcycle graph and the straight skeleton. The results obtained in this section finally lead to a proof for the P-completeness of the straight skeleton of planar straight-line graphs and polygons with holes that is based on the P-completeness of motorcycle graphs due to Eppstein and Erickson [EE99]. The P-completeness of straight skeletons has important practical implications concerning the application of parallel algorithms. The investigations done in Section 3.2 and Section 3.3 are published in [HH11b]. A preliminary and short version was presented in [HH09b]. The results in Section 3.4 have been submitted for publication [HH11a].

在本章中，我们将开发一个用于计算广义摩托车图的实现，该实现可在实践中快速运行。我们从第 3.2 节中对摩托车图中迹线长度的随机考虑开始。结果表明，如果 n 辆摩托车在单位正方形中分布良好，则平均迹线长度约为 $O(1/\sqrt{n})$。这一见解促使我们基于第 3.3 节中的几何哈希来实现摩托车图。运行时实验表明，对于测试的 22 000 个数据集中的大多数，我们的实现 Moca 表现出 O(n log n) 的运行时。在第 3.4 节中，我们将进一步研究摩托车图和直线骨架之间的几何关系。本节中获得的结果最终证明了平面直线图和带孔多边形的直线骨架的 P-完全性，该证明基于 Eppstein 和 Erickson [EE99] 提出的摩托车图的 P-完全性。直线骨架的 P-完全性对于并行算法的应用具有重要的实际意义。第 3.2 节和第 3.3 节中完成的调查发表在 [HH11b] 中。初步的简短版本已在 [HH09b] 中提出。第 3.4 节中的结果已提交发表 [HH11a]。

• [HH09b] S. Huber and M. Held. A Practice-Minded Approach to Computing Motorcycle Graphs. In Proc. 25th Europ. Workshop Comput. Geom., pages 305–308, Brussels, Belgium, Mar 2009

• [HH09b] S. Huber 和 M. Held. 计算摩托车图的一种注重实践的方法。第25届欧洲计算几何研讨会论文集，第305-308页，比利时布鲁塞尔，2009年3月

• [HH11b] S. Huber and M. Held. Motorcycle Graphs: Stochastic Properties Motivate an Efficient Yet Simple Implementation. J. Exp. Algorithmics, 2011. (in press)

• [HH11b] S. Huber 和 M. Held. 摩托车图：随机属性激发了一种高效而简单的实现. J. Exp. Algorithmics, 2011. (出版中)

• [HH11a] S. Huber and M. Held. Approximating a Motorcycle Graph by a Straight Skeleton. 2011. (submitted for publication)

[•] [HH11a] S. Huber 和 M. Held. 通过直线骨架逼近摩托车图. 2011. (已投稿待发表)
3.1 p r i o r a n d r e l a t e d w o r k

3.1 先前及相关工作

3.1.1 Applications of motorcycle graphs and related problems

3.1.1 摩托车图的应用及相关问题

The most prominent application of motorcycle graphs are straight skeletons. However, motorcycle graphs are also related to other geometric problems. Obviously, there exists a strong connection to ray-shooting problems. For instance, Ishaque et al. [IST09] presented an O(n log2 n + m log m) algorithm for m repetitive ray shooting-and-insertion operations in the plane among a set of polygonal obstacles of total size n. The generalized motorcycle graph problem, for which motorcycles need not all start at the same time, solves this problem in the following way: The polygonal obstacles are replaced by according walls and each ray is replaced by a motorcycle. By choosing the start time of the motorcycles such that no two motorcycles drive at the same time, one can simulate repetitive ray-shootings.

摩托车图最突出的应用是直线骨架。然而，摩托车图也与其他几何问题相关。显然，它与射线射击问题存在很强的联系。例如，Ishaque等人[IST09]提出了一个O(n log2 n + m log m)算法，用于在平面上总大小为n的多边形障碍物集合中进行m次重复射线射击和插入操作。广义摩托车图问题（其中摩托车不必都在同一时间启动）通过以下方式解决此问题：多边形障碍物被相应的墙壁代替，每条射线都被摩托车代替。通过选择摩托车的启动时间，使得没有两辆摩托车同时行驶，可以模拟重复的射线射击。

Eppstein and Erickson [EE99] mentioned that the art gallery algorithm by Czyzowicz et al. [CRU89] uses a geometric structure that is related to the motorcycle graph. They consider a set of straight-line segments, where each segment is growing in length and an endpoint of a segment stops propagating when it reaches another segment. The resulting structure can be interpreted as a motorcycle graph: The initial segments are considered as walls and from each endpoint is a motorcycle launched.

Eppstein和Erickson [EE99] 提到，Czyzowicz等人 [CRU89] 的美术馆算法使用了一种与摩托车图相关的几何结构。他们考虑一组直线段，其中每条线段的长度都在增长，并且当线段的端点到达另一条线段时，该端点停止传播。由此产生的结构可以解释为摩托车图：初始线段被视为墙壁，并且从每个端点都发射出一辆摩托车。

Eppstein et al. [EGKT08] consider motorcycle graphs on quadrilateral meshes, which model three-dimensional bodies. The idea is that motorcycles are driving on the edges of the mesh in a discrete manner and a motorcycle stops when it reaches the trace of another motorcycle. It is not exactly the same motorcycle graph problem as introduced by [EE99], but very similar in nature.

Eppstein 等人 [EGKT08] 考虑了四边形网格上的摩托车图，该图对三维物体进行建模。其思想是，摩托车以离散的方式在网格的边上行驶，当摩托车到达另一辆摩托车的轨迹时，它就会停止。这与 [EE99] 提出的摩托车图问题并不完全相同，但在本质上非常相似。

3.1.2 Prior work

3.1.2 前期工作

At the moment, the most efficient motorcycle graph algorithm is due to Cheng and Vigneron [CV07], see Section 1.4.2.4. It computes the motorcycle graph of n motorcycles in O(n√n log n) time. Let us recall the main idea of their algorithm: Among the O(n2) intersections of the tracks of the motorcycles only O(n) of them correspond to an actual crash event. Hence, the goal is to geometrically separate those motorcycles that do not interact. Cheng and Vigneron achieved this goal via the employment of 1/√n-cuttings on the supporting lines of the traces. However, in order to reach the O(n√n log n) runtime complexity, the algorithm inherently relies on the fact that all motorcycles are known a priori such that the 1/√n-cutting can be constructed prior to the simulation of the motorcycles movement. As a consequence, this algorithm is not suitable for the computation of the generalized motorcycle graph presented in Section 2.4.

目前，最有效的摩托车图算法归功于 Cheng 和 Vigneron [CV07]，参见第 1.4.2.4 节。它在 O(n√n log n) 时间内计算 n 辆摩托车的摩托车图。让我们回顾一下他们算法的主要思想：在摩托车轨迹的 O(n2) 个交点中，只有 O(n) 个交点对应于实际的碰撞事件。因此，目标是在几何上分离那些不相互作用的摩托车。Cheng 和 Vigneron 通过在轨迹的支撑线上采用 1/√n-cuttings 实现了这一目标。然而，为了达到 O(n√n log n) 的运行时间复杂度，该算法本质上依赖于所有摩托车都是先验已知的这一事实，以便可以在模拟摩托车运动之前构造 1/√n-cutting。因此，该算法不适用于计算第 2.4 节中介绍的广义摩托车图。

In order to solve the generalized motorcycle graph problem, where motorcycles are allowed to emerge after the simulation started, we are in need of an algorithm that allows the dynamic insertion of motorcycles. Under this requirement, the fastest algorithm is due to Eppstein and Erickson [EE99] and runs in O(n17/11+ϵ) time and space, see Section 1.4.2.3. However, we want to emphasize that this algorithm is not suitable for an actual implementation due to its algorithmic complexity.

为了解决广义摩托车图问题（允许摩托车在模拟开始后出现），我们需要一种允许动态插入摩托车的算法。在这种要求下，最快的算法归功于Eppstein和Erickson [EE99]，其运行时间和空间复杂度为O(n17/11+ϵ)，见第1.4.2.3节。但是，我们要强调的是，由于其算法的复杂性，该算法不适合实际实现。

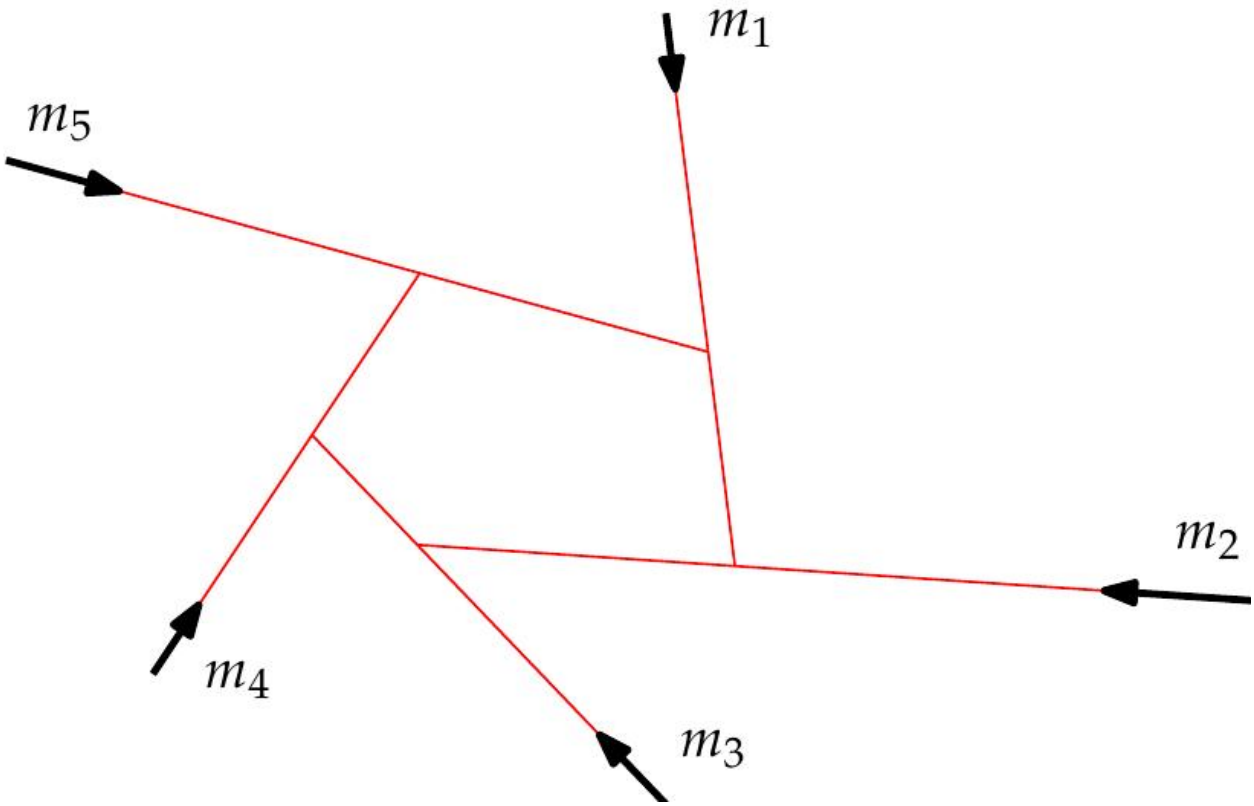


Figure 41: All motorcycles crash into each other.

图 41：所有摩托车互相碰撞。

A straight-forward approach for the actual computation of the motorcycle graph of n motorcycles is to simply simulate their movement in a brute-force manner. We can compute one crash event after the other by pair-wise checks among the motorcycles. By employing a priority queue Q for pending crash events, one obtains an implementation that runs in O(n2 log n) time instead of O(n3). We refer to this algorithm as the standard priority-queue based algorithm.

对于n辆摩托车的摩托车图的实际计算，一个直接的方法是简单地以蛮力方式模拟它们的运动。我们可以通过摩托车之间的两两检查，逐个计算碰撞事件。通过为待处理的碰撞事件采用优先级队列Q，可以获得一个在O(n2 log n)时间内运行的实现，而不是O(n3)。我们将此算法称为基于标准优先级队列的算法。

To sum up, two sub-quadratic algorithms for the computation of motorcycle graphs are known, but only the algorithm by Eppstein and Erickson [EE99] can be used to compute the generalized motorcycle graph problem. Furthermore, no algorithm is known that is suitable for implementation, on one hand, and can be expected to exhibit a significantly sub-quadratic runtime for real-world applications on the other hand.

总而言之，目前已知两种计算摩托车图的亚二次算法，但只有Eppstein和Erickson [EE99]提出的算法可用于计算广义摩托车图问题。此外，目前还没有已知算法既适合实现，另一方面，又能期望在实际应用中表现出显著的亚二次运行时间。

3.1.3 Geometric properties of the motorcycle graph

3.1.3 摩托车图的几何性质

We interpret M(m1, . . . , mn) as a graph and we add the infinite endpoints of the traces of escaped motorcycles to the graph as well. It follows that M(m1, . . . , mn) contains exactly 2n vertices: Each motorcycle contributes a start point and an end point. (If multiple vertices coincide geometrically, we still count them as different vertices.) The number of finite vertices ranges from n to at most 2n. The lower bound is attained if all motorcycles escape. The upper bound is attained if all motorcycles crash into each other, as illustrated in Figure 41.

我们将M(m1, . . . , mn)解释为一个图，并将逃逸摩托车的轨迹的无穷远端点也添加到该图中。由此可知，M(m1, . . . , mn)恰好包含2n个顶点：每辆摩托车贡献一个起点和一个终点。（如果多个顶点在几何上重合，我们仍然将它们视为不同的顶点。）有限顶点的数量范围从n到最多2n。如果所有摩托车都逃逸，则达到下界。如果所有摩托车都相互碰撞，则达到上界，如图41所示。

Lemma 3.1. M(m1, . . . , mn) contains between n and 2n edges. The lower bound is attained if all motorcycles escape. The upper bound is attained if all motorcycles crash.

Lemma 3.1. M(m1, . . . , mn)包含n到2n条边之间。如果所有摩托车都逃脱，则达到下界。如果所有摩托车都坠毁，则达到上界。

Proof. The edge set of M(m1, . . . , mn) results from n motorcycle traces, which are possibly split. Hence, the number of edges is at least n. In order to show the upper bound of 2n, we denote by nc the number of crashed motorcycles and by e the number of edges. M(m1, . . . , mn) contains n + (n − nc) vertices of degree 1 and nc vertices of degree 3. (Note that we obtain n − n^c infinite vertices.) We charge each edge by both of its incident vertices. Considering the total number of charges results in 2e = n + (n − nc) + 3nc = 2n + 2nc ≤ 4n and hence e ≤ 2n. The bound is tight if n = nc.

证明。M(m1, . . . , mn)的边集由n条摩托车轨迹产生，这些轨迹可能被分割。因此，边的数量至少为n。为了展示上限2n，我们用nc表示撞毁的摩托车的数量，用e表示边的数量。M(m1, . . . , mn)包含n + (n − nc)个度为1的顶点和nc个度为3的顶点。（注意，我们得到n − n^c个无限顶点。）我们通过其两个关联顶点对每条边进行收费。考虑总收费数量，得出2e = n + (n − nc) + 3nc = 2n + 2nc ≤ 4n，因此e ≤ 2n。如果n = nc，则该界限是紧的。

The following lemma is given by Eppstein and Erickson [EE99] without a proof. In the following, a pseudo-forest is a graph whose components are pseudo-trees and a pseudo-tree is a connected graph that contains at most one cycle.

以下引理由埃普斯坦和埃里克森 [EE99] 给出，但未提供证明。在下文中，伪森林是指其连通分量为伪树的图，而伪树是指最多包含一个环的连通图。

Lemma 3.2 ([EE99]). M(m1, . . . , mn) is a pseudo-forest.

Lemma 3.2 ([EE99]). M(m1, . . . , mn) 是一个伪森林。

Proof. We denote the number of edges of M(m1, . . . , mn) by e. We may assume that the motorcycle graph M(m1, . . . , mn) is connected; the general case is shown analogously. If M(m1, . . . , mn) is not a pseudo-tree we can remove at least two edges without hurting connectedness. Since a connected graph with 2n vertices contains at least 2n − 1 edges, it follows that e − 2 ≥ 2n − 1. This is a contradiction to Lemma 3.1.

证明。我们用M(m1, . . . , mn)的边数表示为e。我们可以假设摩托车图M(m1, . . . , mn)是连通的；一般情况可以类似地证明。如果M(m1, . . . , mn)不是伪树，我们可以移除至少两条边而不损害连通性。由于一个具有2n个顶点的连通图至少包含2n − 1条边，因此可以得出e − 2 ≥ 2n − 1。这与引理3.1相矛盾。

3.2 s t o c h a s t i c c o n s i d e r a t i o n s o f t h e m o t o r c y c l e g r a p h

3. 2 摩托车图的随机性考虑

3.2.1 Number of intersections of bounded rays

3.2.1 有界射线的交点数量

In order to devise a motorcycle graph algorithm that runs fast in practice, it appears interesting to investigate the average trace length within the motorcycle graph M(m1, . . . , mn) of n motorcycles. Let us recall that the tracks of the motorcycles lead to O(n2) intersections, whereas the traces produce at most O(n) intersections, namely at precisely those points where a motorcycle crashed into the trace of another.

为了设计一个在实践中运行快速的摩托车图算法，研究n辆摩托车的摩托车图M(m1, . . . , mn)中的平均轨迹长度似乎很有趣。让我们回顾一下，摩托车的轨迹会导致O(n2)个交点，而踪迹最多产生O(n)个交点，即恰好在摩托车撞到另一辆摩托车的踪迹上的那些点。

Let us assume for a moment that the motorcycles drive at unit speed, the start points v1, . . . , vn are distributed uniformly in the unit square $[0, 1]^2$ and the ϕ1, . . . , ϕn are distributed uniformly on [0, 2π). Directly computing the expectation of the trace length of a single motorcycle trace appears to be complicated due to the stochastic dependencies among the motorcycle traces. However, it seems to be evident that the mean trace length cannot get too large, because two traces are not allowed to intersect in the interior of each other.

让我们假设摩托车以单位速度行驶，起点v1, . . . , vn在单位正方形$[0, 1]^2$中均匀分布，且ϕ1, . . . , ϕn在[0, 2π)上均匀分布。直接计算单个摩托车轨迹的迹线长度的期望似乎很复杂，因为摩托车轨迹之间存在随机依赖性。然而，平均迹线长度似乎不可能太大，因为两条迹线不允许在彼此的内部相交。

Let us consider a regular √n × √n grid on the unit square. A single motorcycle mi starts at one grid cell and, while it moves, crosses a specific number of other cells. Each cell contains on average one start point of a motorcycle. The probability that mi crosses k cells is falling at least exponentially with k if we would flip a coin to decide whether mi passes the motorcycle that started in each cell entered by mi. This thought experiment

would suggest that a motorcycle does not pass more than O(1) grid cells and, as a consequence, that the average trace length is in O(1/√n). However, a simple coin flip does not take into account the stochastic dependencies among the motorcycles.

让我们考虑单位正方形上的一个规则的√n × √n网格。一辆摩托车mi从一个网格单元格开始，并且在移动时，会穿过特定数量的其他单元格。每个单元格平均包含一个摩托车的起始点。如果我们掷硬币来决定mi是否通过由mi进入的每个单元格中启动的摩托车，那么mi穿过k个单元格的概率至少以指数形式随着k下降。这个思想实验表明，一辆摩托车不会通过超过O(1)个网格单元格，因此，平均轨迹长度在O(1/√n)中。然而，一个简单的掷硬币方法没有考虑到摩托车之间的随机依赖性。

In order to simplify the original question, we reformulate the problem. Instead of asking for the mean trace length of the motorcycles, we ask for the number of intersection points of bounded rays, where the length of the rays is chosen at random according to the probability density function f . The idea is that if we gain some information on the number of intersections than we also gain information on the mean length of the bounded rays as well.

为了简化原始问题，我们对问题进行了重新表述。我们不要求摩托车的平均轨迹长度，而是要求有界射线的交点数量，其中射线的长度是根据概率密度函数f随机选择的。我们的想法是，如果我们获得了关于交点数量的一些信息，那么我们也会获得关于有界射线的平均长度的信息。

Theorem 3.3 ([HH11b]). Let v1, . . . , vn denote n points that are uniformly i.i.d.[1] on the unit square $[0, 1]^2$. Further, by ϕ1, . . . , ϕn we denote n angles i.i.d. on D := {δ1, . . . , δd}, with d ∈ N, such that

定理 3.3 ([HH11b])[NT1]。令 v1, . . . , vn 表示在单位正方形 $[0, 1]^2$ 上均匀独立同分布的 n 个点 [1]。此外，用 ϕ1, . . . , ϕn 表示在 D := {δ1, . . . , δd} 上独立同分布的 n 个角，其中 d ∈ N，使得

3.2 stochastic considerations of the motorcycle graph 83

δi ∈ [0, 2π) occurs with probability pi, where ∑i pi = 1. Next, let L1, . . . , Ln be i.i.d. on [0, 0.2] according to a probability density function f .

δi ∈ [0, 2π) 以概率 pi 发生，其中 ∑i pi = 1。接下来，设 L1, . . . , Ln 是在 [0, 0.2] 上根据概率密度函数 f 的独立同分布。

For each i ∈ {1, 2, . . . , n} consider a bounded ray Ti ⊂ R2 which starts at vi, has direction angle ϕi and length Li. We denote by I = ∑in=2 1T1∩Ti□=□ the number of intersections of T1 with T2, . . . , Tn, where 1P denotes the indicator function of the predicate P. Then

对于每个 i ∈ {1, 2, . . . , n}，考虑一个有界射线 Ti ⊂ R2，它起始于 vi，具有方向角 ϕi 和长度 Li。我们用 I = ∑in=2 1T1∩Ti□=□ 表示 T1 与 T2, . . . , Tn 的交点数，其中 1P 表示谓词 P 的指示函数。那么

$$\Delta 25 . E [L 1] 2 (n − 1)$$

≤
E
[
I
]
≤
Δ
.
E
[
L
1
]
2
(
n
−
1
)
,

(3.1)

holds, where $\Delta := \sum_{i,j=1} p_i p_j |\sin(\delta_i - \delta_j)|$. Furthermore, for $\Delta > 0$ we obtain

成立，其中 $\Delta := \sum_{i,j=1} p_i p_j |\sin(\delta_i - \delta_j)|$。此外，对于 $\Delta > 0$，我们得到

E
[
I

]
∈
Θ
□
n
E
[
L
1
]
2
□
.

(3.2)
(Distributing L1, . . . , Ln on the interval [0, 0.2] is just a technicality that simplifies the following proof of the theorem.)

（在区间[NT1][[/[NT1]0，0.2[NT5]][/[NT5]上分布L1, . . . , Ln只是一个技术细节，它简化了以下定理的证明。）

Proof. We assume that pi > 0 for all 1 ≤ i ≤ n. Hence, ∆ is zero if and only if δi − δj ∈ πZ for all 1 ≤ i, j ≤ d. The latter condition means that the supporting lines of the bounded rays are parallel. Hence, two rays intersect with probability zero and the claim of the theorem is trivial. So let us assume ∆ > 0. The law of total expectation yields

证明。我们假设对于所有 1 ≤ i ≤ n，pi > 0。因此，∆ 为零当且仅当对于所有 1 ≤ i, j ≤ d，δi − δj ∈ πZ。后一个条件意味着有界射线的支撑线是平行的。因此，两条射线相交的概率为零，并且该定理的结论是微不足道的。所以让我们假设 ∆ > 0。全期望定律得出

E
[
I
]

=

d

Σ

i

=

1

P

(

ϕ

1

=

δ

i

)

E

[

I

|

ϕ

1

=

δ

i

]

=

d

Σ

i

=

1

p

i

E

[

I

|

ϕ

1

=

δ

i

]

.

(3.3)

Consider the ray T1 fixed. In order to have a ray Tk intersect T1 the start point vk of Tk needs to start in a certain area, whose shape depends on the direction ϕk of Tk. For some arbitrary direction ϕ we denote this area by a parallelogram Sϕ that is hinged at v1, see Figure 42:

考虑射线T1固定。为了使射线Tk与T1相交，Tk的起始点vk需要从某个区域开始，该区域的形状取决于Tk的方向ϕk。对于某个任意方向ϕ，我们将该区域表示为一个以v1为铰链的平行四边形Sϕ，参见图42：

S

ϕ

:

=

□

v
1
+
a
□
cos
ϕ
1
sin
ϕ
1
□
−
b
□
cos
ϕ
sin
ϕ
□
:
a
∈
[
0,

L
1
]
,
b
∈
[
0,
0.2
]
□

.
The mapping

该映射

F
:
R
2
→
R
2
:
v
7
→
□

cos
ϕ
1
sin
ϕ
1
−
sin
ϕ
1
cos
ϕ
1
□
.
(
v
−
v
1
)
models the translation of v1 to the origin and subsequent clockwise rotation by the angle ϕ1. Hence, F(T1) starts at the origin and points rightwards, see Figure 42.

模拟了v1到原点的平移以及随后顺时针旋转角度ϕ1的过程。因此，F(T1)从原点开始并指向右侧，参见图[NT3]42]。

For a ray Tk to intersect T1, it is necessary that the start point vk lies is in Sϕk . (Keep in mind that the lengths are restricted to [0, 0.2] and Tk has the direction angle ϕk.) We denote by (xi′, yi′) := vi′ := Fvi the translated start points vi, with 1 ≤ i ≤ n. Note that Tk

intersects T1 if and only if F(Tk) intersects F(T1). However, F(Tk) intersects F(T1) if and only if v′ k ∈ F(Sϕk) and Lk| sin(ϕk − ϕ1)| ≥ |y′ k|. (The latter condition basically states that Tk is long enough in order to intersect T1.)

对于射线Tk与T1相交，起点vk必须位于Sϕk内。（请记住，长度被限制在[0，0.2]内，并且Tk的方向角为ϕk.)。）我们用(xi′, yi′) := vi′ := Fvi表示平移后的起点vi，其中1 ≤ i ≤ n。注意，Tk与T1相交当且仅当F(Tk)与F(T1)相交。然而，F(Tk)与F(T1)相交当且仅当v′ k ∈ F(Sϕk)且Lk| sin(ϕk − ϕ1)| ≥ |y′ k|。（后一个条件基本上说明Tk足够长才能与T1.)相交。）
We note that Sϕ ⊂ [0, 1]2 holds for all ϕ1 ∈ D and ϕ ∈ [0, 2π) only if v1 ∈ [0.4, 0.6]2. Hence, for any δi ∈ D it follows that

我们注意到，当且仅当v1 ∈ [0.4, 0.6]2时，对于所有ϕ1 ∈ D和φ ∈ [0, 2π)，Sϕ ⊂ [0, 1]2成立。因此，对于任何δi ∈ D，可以得出

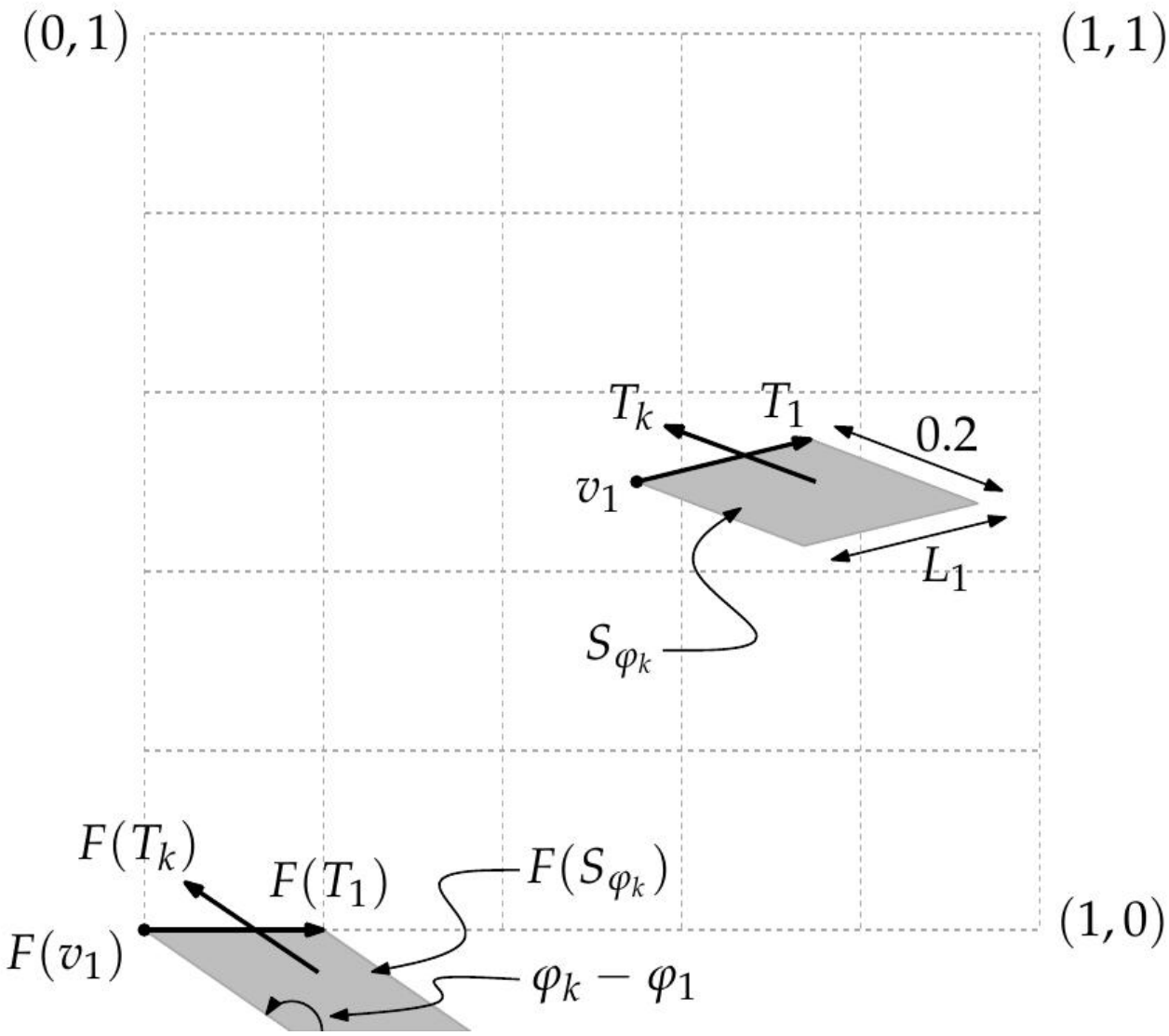


Figure 42: The big 5 × 5 grid illustrates $[0, 1]^2$ with the origin at the left bottom point. We see that λ(Sϕk) = L1 · 0.2 · | sin(ϕk − ϕ1)|, where λ denotes the Lebesgue measure.

图 42：大的 5 × 5 网格展示了 $[0, 1]^2$，原点位于左下角。我们看到 λ(Sϕk) = L1 · 0.2 · | sin(ϕk − ϕ1)|，其中 λ 表示勒贝格测度。

$$
E
[
I
|
\phi
1
=
\delta
i
]
\leq
E
[
I
|
\phi
1
=
\delta
i
,
V
$$

$$1 \in [0.4, 0.6]^2]$$.

On the other hand, by the law of total expectation we obtain

另一方面，根据全期望定律，我们得到

$$P(V1 \in [0.4, 0.6]^2)$$.

$$E$$

$$
\begin{bmatrix}
\mathbf{I} \\
| \\
\phi_1 = \delta_i, \mathbf{V}_1 \in [0.4, 0.6]_2
\end{bmatrix} \leq \mathbf{E}\begin{bmatrix}
\mathbf{I} \\
|
\end{bmatrix}
$$

ϕ
1
=
δ
i
]
,
and therefore

因此

1
25
.

E
[
I
|
A
i
]
≤
E
[
I

|
ϕ
1
=
δ
i
]
≤
1
.
E
[
I
|
A
i
]
,
where Ai denotes the event that ϕ1 = δi and v1 ∈ [0.4, 0.6]2. Summing up over all i according to Equation (3.3) gives

其中Ai表示事件ϕ1 = δi和v1 ∈ [0.4, 0.6]2。根据公式（3.3）对所有i求和得出

1
25
.
d
Σ

i

=

1

p

i

E

[

I

|

A

i

]

≤

E

[

I

]

≤

1

.

d

Σ

i

=

1

p

i

E

[

I

|

A

i

]

.

(3.4)

In the next step, we analyze E[I|Ai]. Let Ij denote the number of intersections caused by rays having a direction angle δj. Hence ∑dj=1 Ij = I. We further denote by Bi,j,m ⊆ Ai those events of Ai, where exactly m rays point to direction δj. Note that each ray causes intersections independently to each other. Therefore, we can separate the cases according to the distribution of the direction angles ϕ2, . . . , ϕn, which is multinomial:

在下一步中，我们分析E[I|Ai]。令Ij表示方向角为δj的光线所造成的交点数。因此，∑dj=1 Ij = I。我们进一步用Bi,j,m ⊆ Ai表示Ai中那些恰好有m条光线指向方向δj的事件。请注意，每条光线彼此独立地产生交点。因此，我们可以根据方向角ϕ2, . . . , ϕn的分布来分离情况，这是一个多项式分布：

E

[

I

|

$$A_i] = d\sum_{j=1} E[I_j|A_i] = d\sum_{j=1} 25$$

.

Z

[

0.4,0.6

]

2

Z

0.2

0

∑

n

1

+

.

.

.

+

n

d

=

n

−

1

□

n

−

1

n

1

,

.

.

.

,

n

d

□

p

n

1

1

.

.

.

p

n

d

d

E

[

I
j
|
B
i
,
j
,
n
j
]
d
f
(
L
1
)
d
v
1
.

(3.5)

Next, we analyze E[Ij|Bi,j,m]. We observe that E[Ij|Ai] is zero for i = j. Hence, we may assume i □= j in the sequel. Recall that we are asking for the expected number of intersections of T1 with m rays that point in direction δj and which are distributed

independently. Hence, we may assume that the rays T2, . . . , Tm+1 are driving in direction δj. Recall that Tk intersects T1 only if vk ∈ Sδj . Denoting by λ the Lebesgue measure, we obtain

接下来，我们分析E[Ij|Bi,j,m]。我们观察到，当 i = j 时，E[Ij|Ai]为零。因此，在下文中，我们可以假设i □= j。回想一下，我们正在询问T1与m条指向δj方向且独立分布的光线的预期交点数。因此，我们可以假设光线T2, . . . , Tm+1沿δj方向传播。回想一下，Tk仅当vk ∈ Sδj时才与T1相交。用λ表示勒贝格测度，我们得到

$$E[I_j|B_{i,j,m}] = m\lambda(S_{\delta_j}) = 0 \square$$

m

l

□

λ

(

S

δ

j

)

l

(

1

−

λ

(

S

δ

j

)

)

m

−

l

E
[
I
j
|
A
i
,
j
,
l
]
,
where Ai,j,l ⊆ Bi,j,m denotes the event that exactly l of the rays of Bi,j,m start within Sδj .

其中 Ai,j,l ⊆ Bi,j,m 表示事件：Bi,j,m 的射线中恰好有 l 条起始于 Sδj 内。

We now resolve E[Ij|Ai,j,l]. W.l.o.g. assume that T2, . . . , Tl+1 start in Sδj . Recall the notation (xk′ , y′ k) := v′ k := F(vk) and recall that we may assume i □= j, since E[Ij|Ai] is zero for i = j. Since every ray causes intersections independently, we get

现在我们求解E[Ij|Ai,j,l]。不失一般性，假设T2, . . . , Tl+1起始于Sδj。回顾记号(xk′ , y′ k) := v′ k := F(vk)，并回顾我们可以假设i □= j，因为对于i = j，E[Ij|Ai]为零。由于每条射线独立地引起交点，我们得到

E
[
I
j

|
A
i
,
j
,
l
]
=
l
+
1
Σ
k
=
2
1
λ
(
S
δ
j
)
z
S

δ

j

Z

0.2

0

1

T

k

∩

T

1

□

=

∅

d

f

(

L

k

)

d

V

k

=

I

1

λ

(

S

δ

j

)

Z

F

S

δ

j

Z

0.2

0

1

L

2

|

sin

(

δ

i

–

δ

j

)

|

≥

|

y

′

2

|

d

f

(

L

2

)

d

v

′

2

=

l

1

λ

(

S
δ
j
)
L
1
Z
0.2
|
sin
(
δ
i
–
δ
j
)
|
0
P
(
L
2
|
sin
(

δ

i

−

δ

j

)

|

≥

y

′

2

)

d

y

′

2

.

Next, we substitute y2′ by z := y2′ /| sin(δj − δi)| and get

接下来，我们用z := y2′ /| sin(δj − δi)|替换y2′，得到

E

[

I

j

|

A

i

,

j

,

l

]

=

I

|

sin

(

δ

j

−

δ

i

)

|

λ

(

S

δ

j

)

L

1

Z

0.2

0

P

(

L

2

≥

Z

)

d

Z

=

I

|

sin

(

δ

j

$$
-
\delta
i
)
|
L
1
\lambda
(
S
\delta
j
)
E
[
L
2
]
=
5
I
E
$$

[
L
1
]
.

(3.6)
Since ∑n k=0 (nk)pk(1 − p)n−k equals 1, we can plug the last result into the expression for E[lj|Bi,j,m] and get

由于 ∑n k=0 (nk)pk(1 − p)n−k 等于 1，我们可以将最后的结果代入 E[lj|Bi,j,m] 的表达式中，得到

E
[
l
j
|
B
i
,
j
,
m
]
=
m

$$\sum_{l=0}^{m} l \, \lambda(S_{\delta j}) \, l(1-\lambda(S_{\delta j})$$

)
)
m
–
I
5
I
E
[
L
1
]
=
5
E
[
L
1
]
λ
(
S

δ

j

)

m

.

m

Σ

l

=

1

□

m

–

1

l

–

1

□

λ

(

S

δ

j

)
I
–
1
(
1
–
λ
(
S
δ
j
)
)
m
–
1
–
(
I
–
1
)
=
5

$$E[L_1]_{\lambda(S_{\delta_j})}m = mE[L_1]L_1$$

|
sin
(
δ
i
−
δ
j
)
|
.
In the final step, we plug this result into Equation (3.5) and obtain

在最后一步，我们将此结果代入公式 (3.5) 并得到

E
[
I
|
A
i
]
=
d
Σ

j

=

1

25

Z

[

0.4,0.6

]

2

Z

0.2

0

Σ

n

1

+

.

.

.

+

n

d

=

n

−

1

□

n

–

1

n

1

,

.

.

.

,

n

d

□

p

n

1

1

.

.

.

p

n

d

d

n
j
E
[
L
1
]
L
1
|
sin
(
δ
i
–
δ
j
)
|
d
f
(

L

1

)

d

V

1

=

25

Z

[

0.4,0.6

]

2

d

V

1

.

E

[

L

1

]

Z

0.2

$$
\begin{aligned}
&L_1 \\
&d f(L_1). \\
&d\Sigma_{j=1} \left| \sin(\delta i - \delta j) \right|
\end{aligned}
$$

|

Σ

n

1

+

.

.

.

+

n

d

=

n

−

1

n

j

□

n

−

1

n

1

,

.

.

.

,

n

d

□

p

n

1

1

.

.

.

p

n

d

d

.

(3.7

)

Next we use

接下来我们使用

n

j

□

n

–

1

n

1

,

.

.

.

,

n

d

□

=

(

n

–

1

)

□

n

–

2

n

1

,

.

.

.

,

n

j

–

1,

.

.

.

n

d

□

and therefore see that

因此可见

$$\sum_{n}^{1} + \cdots + n_d = n$$

–

1

n

j

□

n

–

1

n

1

,

.

.

.

,

n

d

□

p

n

1

1

.

.

.

p

n

d

d

=

(

n

–

1

)

p

j

.

Finally, by E[L1] = R00.2 L1 d f (L1), it follows that

最后，根据E[L1] = R00.2 L1 d f (L1)，可以得出

E

[

I

|

A

i

]

=

E

[

L

1

]

2

(

n

−

1

)

.

d

∑

j

=

1

p

j

|

sin

(

δ

i

−

δ

j

)

|

.

(3.8)
Using this result in Equation (3.4) finally proves the assertions of the theorem.

将此结果应用于方程 (3.4) 最终证明了定理的断言。

Choosing the lengths L1, . . . , Ln on the interval [0, 0.2] was a technical twist. It allowed us to assume that if a bounded ray T1 starts in $[0.4, 0.6]^2$ and is intersected by another bounded ray T2 then T2 started definitely in $[0, 1]^2$, because the start points of T1 and T2 have a distance of at most 0.4. Therefore, it holds that

在区间[NT1][0, 0.2]上选择长度L1, . . . , Ln是一个技术上的技巧。这使得我们可以假设，如果一条有界射线T1起始于$[0.4, 0.6]^2$，并且被另一条有界射线T2相交，那么T2一定起始于$[0, 1]^2$，因为T1和T2的起始点之间的距离最多为0.4。因此，下式成立：

1

.

E

[

I

|

A

]

≤

E

[

I

]

≤

E

[

I

|

A

]

,
where A denotes the event that T1 started in $[0.4, 0.6]^2$. Note that the left inequality follows by the law of total expectation. In order to prove Theorem 3.3 it remained to show that $E[I|A] = \Delta \cdot E[L1]2(n - 1)$. However, if we distribute L1, . . . , Ln on an interval [0, ϵ], with any positive $\epsilon < 0.25$, the argument from above easily extends to

其中 A 表示 T1 在 $[0.4, 0.6]^2$ 开始的事件。请注意，左侧不等式由全期望定律得出。为了证明定理 3.3，仍然需要证明 $E[I|A] = \Delta \cdot E[L1]2(n - 1)$。然而，如果我们将 L1, . . . , Ln 分布在区间 [0, ε] 上，其中任何正数 $\varepsilon < 0.25$，则上述论证很容易扩展到

(

1

−

4

ϵ

)

2

.

E

[

I

|

A

]
≤
E
[
I
]
≤
E
[
I
|
A
]
,
where A denotes the event that v1 ∈ [2ϵ, 1 − 2ϵ]2. Note that the remainder of the proof of Theorem 3.3 is not affected by the above generalization. As a consequence, Theorem 3.3 can actually be generalized to

其中A表示事件v1 ∈ [2ϵ, 1 − 2ϵ]2。请注意，定理3.3的剩余证明不受上述推广的影响。因此，定理3.3实际上可以推广到

(
1
−
4
ϵ

$$
)
2
\Delta
.
E
[
L
1
]
2
(
n
-
1
)
\leq
E
[
I
]
\leq
\Delta
.
$$

E

[

L

1

]

2

(

n

−

1

)

.

(3.9)

3.2.2 Implications to the motorcycle graph

3.2.2 对摩托车图的影响

Consider n random motorcycles m1, . . . , mn that drive at unit speed and where the start points v1, . . . , vn are chosen uniformly from the unit square $[0, 1]^2$ and the direction angles ϕ1, . . . , ϕn are chosen from a set D := {δ1, . . . , δd}, where δi ∈ [0, 2π) occurs with probability pi, as in Theorem 3.3. After generating n random motorcycles as described above, we compute the motorcycle graph M(m1, . . . , mn) and record all the trace lengths. We can repeat this experiment a number of times and keep on recording the trace lengths. The samples recorded can be used to obtain an approximation fˆ of the density of the trace lengths of a motorcycle graph with n motorcycles. Using the approximate density fˆ in Theorem 3.3 establishes the relation (3.9) between the expected number of intersections E[I] and the mean trace length E[L1] according to the approximate density function fˆ. Unfortunately, both E[I] and E[L1] are unknown.

考虑 n 辆随机摩托车 m1, . . . , mn，它们以单位速度行驶，起点 v1, . . . , vn 从单位正方形 $[0, 1]^2$ 中均匀选择，方向角 ϕ1, . . . , ϕn 从集合 D := {δ1, . . . , δd} 中选择，其中 δi ∈ [0, 2π) 以概率 pi 出现，如定理 3.3 所示。在生成如上所述的 n 辆随机摩托车后，我们计算摩托车图 M(m1, . . . , mn) 并记录所有轨迹长度。我们可以重复这个实验多次，并持续记录轨

迹长度。记录的样本可用于获得具有 n 辆摩托车的摩托车图的轨迹长度密度的近似值 f^。在定理 3.3 中使用近似密度 f^ 建立了期望交点数 E[I] 和平均轨迹长度 E[L1] 之间根据近似密度函数 f^ 的关系 (3.9)。不幸的是，E[I] 和 E[L1] 都是未知的。
However, for increasingly larger values of n the vast majority of motorcycles does not reach the boundary of $[0, 1]^2$, but crashes against other traces. Hence, as the number n of motorcycles increases, the trace lengths shrink in the average case[2], and for sufficiently large n the vast majority of motorcycles can be expected to have a trace length less than some constant ϵ smaller than 0.25. Since there are at most n crashes, a motorcycle trace may be assumed to intersect two other traces on average: the motorcycle itself crashes into another trace and a second motorcycle crashes into the considered trace. This suggests E[I] = 2, which can also be verified experimentally. (Actually, $E[I] \in O(1)$ would suffice for our subsequent runtime analysis of our algorithm in Section 3.3.)

然而，对于越来越大的n值，绝大多数摩托车无法到达$[0，1]^2$的边界，而是与其他轨迹发生碰撞。因此，随着摩托车数量n的增加，轨迹长度在平均情况下会缩短[2]，并且对于足够大的n，可以预期绝大多数摩托车的轨迹长度小于某个小于0.25的常数ϵ。由于最多有n次碰撞，因此可以假设摩托车轨迹平均与其他两条轨迹相交：摩托车本身撞到另一条轨迹，而第二辆摩托车撞到所考虑的轨迹。这表明 E[I] = 2，这也可以通过实验验证。（实际上，$E[I] \in O(1)$ 对于我们在第 3.3 节中对算法的后续运行时分析来说就足够了。）

Plugging E[I] = 2 and a small ϵ in the inequality (3.9) suggests the following approximation for the mean trace length:

将 E[I] = 2 和一个小的 ϵ 代入不等式 (3.9) 中，可以得到以下平均迹线长度的近似值：

$$E[L1] \approx s\,2(n$$

−
1
)
Σ
d
i
,
j
=
1
p
i
p
j
|
sin
(
δ
i
−
δ
j
)

(3.10)

Of course, the assumption that L1, . . . , Ln are independently distributed is not justified for the actual motorcycle graph problem. However, in Section 3.3.3, we are able to substantiate this approximative formula for the mean trace length of motorcycle graphs by providing sound experimental evidence.

当然，对于实际的摩托车图问题，L1, . . . , Ln是独立分布的假设是不合理的。然而，在第3.3.3节中，我们通过提供可靠的实验证据来证实摩托车图平均轨迹长度的这个近似公式。

3.3 a s i m p l e a n d p r a c t i c e - m i n d e d i m p l e m e n t a t i o n

3.3 一个简单且注重实践的实现

In order to compute the straight skeleton S (G) of a planar straight-line graph G by using Bone, it is vital that we can compute the motorcycle graph M(G) fast in practice. The algorithm by Eppstein and Erickson [EE99] would allow us to compute M(G), but the algorithm is too complicated to be implemented. The algorithm by Cheng and Vigneron [CV07] is easier, but still complicated to implement — e. g., we would need to implement an algorithm for the 1/√n-cutting — and it needs to know all motorcycles a priori. The standard priority-queue based algorithm is trivial to implement, but exhibits an O(n2 log n) runtime in practice. For our purposes, we seek a motorcycle graph implementation that is simple enough to be implemented, runs fast in practice, and which supports the dynamic insertion of new motorcycles.

为了使用 Bone 计算平面直线图 G 的直线骨架 S (G)，至关重要的是我们能够在实践中快速计算摩托车图 M(G)。Eppstein 和 Erickson [EE99] 的算法允许我们计算 M(G)，但该算法过于复杂而无法实现。Cheng 和 Vigneron [CV07] 的算法更容易理解，但实现起来仍然很复杂——例如，我们需要实现一个用于 1/√n-cutting 的算法——并且它需要事先知道所有的摩托车。基于标准优先级队列的算法实现起来很简单，但在实践中表现出 O(n2 log n) 的运行时间。就我们的目的而言，我们寻求一种摩托车图实现，它足够简单易于实现，在实践中运行速度快，并且支持动态插入新的摩托车。

Our approach is to take the algorithm of Cheng and Vigneron [CV07], drop the arrangements on each cutting cell and to replace the 1/√n-cutting by a regular √n × √n grid. In other words, we apply geometric hashing to the standard priority-queue based algorithm. The motivation for our approach is a simple trade-off: we lose the deterministic O(n√n log n) time complexity, but instead gain an algorithm that (i) is easy to implement and (ii) runs in O(n log n) time if the motorcycles are sufficiently uniformly distributed.

我们的方法是采用 Cheng 和 Vigneron [CV07] 的算法，去掉每个切割单元上的排列，并用规则的 √n × √n 网格代替 1/√n-cutting。换句话说，我们将几何哈希应用于基于标准优先级队列的算法。我们采用这种方法的动机是一个简单的权衡：我们失去了确定性的 O(n√n log

n) 时间复杂度，但取而代之的是获得了一种算法，该算法 (i) 易于实现，并且 (ii) 如果摩托车足够均匀地分布，则以 O(n log n) 时间运行。

3.3.1 Details of the algorithm

3.3.1 算法细节

The input to our algorithm consists of a set M = {m1, . . . , mn} of motorcycles and a set W = {w1, . . . , wu} of rigid walls. The motorcycles need not all be known a-priori. A wall is modeled as a straight-line segment and a motorcycle mi is given by a start point vi, a speed vector si and a start time ti∗ ∈ [0, ∞). Due to practical numerical advantages, we scale the input such that the bounding box of the start points is a proper subset of the unit square $[0, 1]^2$.

我们算法的输入包括一个摩托车集合M = {m1, . . . , mn}和一个刚性墙集合W = {w1, . . . , wu}。并非所有的摩托车都需要事先知道。墙被建模为一条直线段，而摩托车mi由一个起始点vi、一个速度向量si和一个起始时间ti∗ ∈ [0, ∞)给出。由于实际数值上的优势，我们对输入进行缩放，使得起始点的边界框是单位正方形$[0, 1]^2$的真子集。

For the matter of simplicity, we first restrict our computation of the motorcycle graph to the unit square $[0, 1]^2$. However, this restriction can be waived easily, see Section 3.3.4. (The restriction of the computation to $[0, 1]^2$ can be enforced within our framework by adding four dummy walls that form the boundary of [0, 1]2.)

为了简化起见，我们首先将摩托车图的计算限制在单位正方形$[0, 1]^2$内。然而，这个限制可以很容易地解除，参见第3.3.4节。（通过添加四个形成[0, 1]2.)边界的虚拟墙，可以在我们的框架内强制将计算限制在$[0, 1]^2$内。）

Our algorithm maintains two geometric hashes, HM and HW, which form regular √n × √n grids on $[0, 1]^2$. While HM keeps track of the motorcycles, HW contains the walls of W. We use HW in order to determine whether a motorcycle crashed against a wall. However, since we consider walls to be rigid, we could have employed more efficient ray-shooting algorithms in terms of worst-case complexity. For the matter of simplicity, we choose the geometric hash for our implementation.

我们的算法维护两个几何哈希，HM 和 HW，它们在 $[0, 1]^2$ 上形成规则的 √n × √n 网格。虽然 HM 跟踪摩托车，但 HW 包含 W 的墙壁。我们使用 HW 以确定摩托车是否撞到墙壁。然而，由于我们认为墙壁是刚性的，因此我们可以使用更有效的光线投射算法，就最坏情况复杂度而言。为了简单起见，我们选择几何哈希用于我们的实现。

The basic algorithm is a discrete event simulation of the movement of the motorcycles with two types of events: crash events and switch events. A crash event indicates that a motorcycle crashes against another motorcycle or a wall, and a switch event occurs when a motorcycle leaves one grid cell and enters a neighboring one. All events pending are kept in a priority queue Q. Furthermore, for every motorcycle mi, we maintain a balanced binary search tree C[mi] that contains potential future crash events of the motorcycle mi.

基本算法是对摩托车运动的离散事件模拟，包含两种类型的事件：碰撞事件和切换事件。碰撞事件表示一辆摩托车撞向另一辆摩托车或墙壁，而切换事件发生在摩托车离开一个网格单元并进入相邻单元时。所有待处理的事件都保存在优先级队列 Q 中。此外，对于每辆摩托车 mi，我们维护一个平衡二叉搜索树 C[mi]，其中包含摩托车 mi 的潜在未来碰撞事件。
The algorithm starts with filling HW with all walls of W and then invokes insertMc(m) for each m ∈ M, which inserts a new motorcycle m to our data structures. The main loop of the algorithm extracts one event e from Q after the other and processes them by calling handle(e), depending on the actual type of the event e. If a newly emerging motorcycle m should be inserted at any time of computation then insertMc(m) is called. The procedures insertMc and handle are described in the sequel:

算法首先用W的所有墙壁填充HW，然后对每个m ∈ M调用insertMc(m)，这会将新的摩托车m插入到我们的数据结构中。算法的主循环从Q中依次提取一个事件e，并通过调用handle(e)来处理它们，具体取决于事件e的实际类型。如果在计算的任何时候应该插入新出现的摩托车m，则调用insertMc(m)。过程insertMc和handle将在下文描述：

• insertMc(motorcycle m): We first create an empty binary search tree C[m] and then insert a switch event for m into Q. The occurrence time of the switch event is set to the start time of m.

• insertMc(摩托车 m)：我们首先创建一个空的二叉搜索树 C[m]，然后将 m 的一个切换事件插入到 Q 中。该切换事件的发生时间被设置为 m 的开始时间。

• handle(switch event e of the motorcycle m): We register m at the cell that m entered and add the subsequent switch event of m to Q, if one exists. Then we check for a potential crash against a wall in the current cell and add the earliest one, if existing, as a crash event to Q. We clear C[m] and for every other motorcycle m′ registered in the current cell, we check for an intersection of the tracks of m and m′. For each such intersection, we add a corresponding crash event into C[m] if m′ reaches the intersection before m, and into C[m′] for the dual case. Note that if we add an event into C[m′] that ends up being the earliest in C[m′] then we have to update Q accordingly. Finally, we add the earliest crash event of C[m] into Q.

• handle(摩托车 m 的切换事件 e): 我们在 m 进入的单元格注册 m，并将 m 的后续切换事件添加到 Q（如果存在）。然后，我们检查当前单元格中是否存在与墙壁潜在的碰撞，并将最早的碰撞事件（如果存在）作为碰撞事件添加到 Q。我们清除 C[m]，并对于当前单元格中注册的每辆其他摩托车 m′，我们检查 m 和 m′ 的轨迹是否相交。对于每个这样的交点，如果 m′ 在 m 之前到达交点，我们将相应的碰撞事件添加到 C[m] 中；对于对偶情况，则添加到 C[m′] 中。请注意，如果我们将一个事件添加到 C[m′] 中，而该事件最终成为 C[m′] 中最早的事件，那么我们必须相应地更新 Q。最后，我们将 C[m] 中最早的碰撞事件添加到 Q 中。

• handle(crash event e of the motorcycle m): First, we mark the motorcycle m as crashed and clear C[m]. Note that the trace of m ends at the corresponding crash point. Secondly, we remove the remaining switch event of m from Q. (Alternatively, we could leave the

event in Q, but check at each switch event whether the current event is still valid.) Then we clean up interactions with other motorcycles m′ in the current grid cell: We remove from Q all crash events, where m is involved and which got invalid, because it turned out that m will not reach the location of the potential crash event. Likewise, if C[m′] contains such an invalid crash event then it is removed as well.
• 处理(摩托车 m 的碰撞事件 e): 首先，我们将摩托车 m 标记为已碰撞，并清除 C[m]。请注意，m 的轨迹在相应的碰撞点结束。其次，我们从 Q 中移除 m 剩余的切换事件。（或者，我们可以将该事件留在 Q 中，但在每个切换事件中检查当前事件是否仍然有效。）然后，我们清理与当前网格单元中其他摩托车 m′ 的交互：我们从 Q 中移除所有碰撞事件，其中涉及 m，并且这些事件由于 m 不会到达潜在碰撞事件的位置而变得无效。同样，如果 C[m′] 包含此类无效的碰撞事件，则也会将其移除。
3.3.2 Runtime analysis

3.3.2 运行时分析

In the subsequent analysis, we ignore the influence of the wall handling and concentrate on the computation of the motorcycle graph only. The procedure insertMc is called exactly n times, which takes O(n log n) time in total. A single crash event or switch event of a motorcycle is handled in O(k log n) time, where k ∈ O(n) denotes the number of motorcycles in the cell affected. Note that we can remove an element of Q in O(log n) time if we have a pointer to the element and if we use, for instance, a maximizing heap to implement Q.

在随后的分析中，我们忽略了墙壁处理的影响，仅专注于摩托车图的计算。过程insertMc被精确地调用n次，总共花费O(n log n)时间。一个摩托车的单次碰撞事件或切换事件在O(k log n)时间内处理，其中k ∈ O(n)表示受影响单元格中的摩托车数量。请注意，如果我们有一个指向Q元素的指针，并且如果我们使用例如最大化堆来实现Q，我们可以在O(log n)时间内删除Q的一个元素。

In total we have O(n) crash events and at most O(n√n) switch events. Hence, in the worst case, our algorithm runs in O(n2√n log n) time. However, it seems very unlikely that the worst case actually happens: it would require that Ω(n) motorcycles drive across Ω(√n) common grid cells. Hence, those Ω(n) motorcycles drive virtually parallel along a long strip that is only O(1/√n) units thick and, moreover, no other motorcycle is allowed to cross this strip, until the motorcycles crossed a constant fraction of the whole grid.

总共有O(n)次碰撞事件，以及最多O(n√n)次切换事件。因此，在最坏的情况下，我们的算法运行时间为O(n2√n log n)。然而，最坏的情况似乎不太可能发生：它需要Ω(n)辆摩托车驶过Ω(√n)个公共网格单元。因此，这些Ω(n)辆摩托车实际上是沿着一条只有O(1/√n)单位厚度的长条几乎平行地行驶，而且，在这些摩托车穿过整个网格的恒定比例之前，不允许其他摩托车穿过这条长条。
As we learned in Section 3.2, a motorcycle is visiting O(1) grid cells on average, if the motorcycles are distributed uniformly enough. Consequently, a single grid cell is occupied by O(1) motorcycles on average. Again, the initialization consumes O(n log n) time in total. However, now a single crash event or switch event is handled in O(log n) time in the average case. Still, we have O(n) crash events but in the mean observe only O(1) switch

events per motorcycle. Summarizing, we may expect a runtime of O(n log n) for sufficiently uniformly distributed input, as motivated by Section 3.2. We provide sound experimental evidence in Section 3.3.3 that underpins our arguments for an O(n log n) runtime for uniformly distributed start points, but also demonstrates an O(n log n) runtime for most of the real-world input, where start points are not necessarily distributed uniformly.

正如我们在第 3.2 节中所了解的，如果摩托车分布得足够均匀，那么平均而言，一辆摩托车会访问 O(1) 个网格单元。因此，平均而言，单个网格单元被 O(1) 辆摩托车占据。同样，初始化总共消耗 O(n log n) 时间。然而，现在在平均情况下，单个碰撞事件或切换事件在 O(log n) 时间内处理。尽管如此，我们有 O(n) 个碰撞事件，但在平均情况下，每辆摩托车仅观察到 O(1) 个切换事件。总而言之，正如第 3.2 节所论证的，对于充分均匀分布的输入，我们可以预期运行时间为 O(n log n)。我们在第 3.3.3 节中提供了可靠的实验证据，支持我们关于均匀分布的起始点运行时间为 O(n log n) 的论点，同时也证明了对于大多数现实世界的输入，运行时间为 O(n log n)，其中起始点不一定是均匀分布的。
3.3.3 Experimental results and runtime statistics

3.3.3 实验结果与运行时统计

Our motorcycle code is called Moca[3]. It is implemented in C++ and uses the STL for common data structures like queues, red-black trees and priority queues. All geometric computations are based on ordinary IEEE 745 double-precision floating-point arithmetic. Moca provides runtime options for a posteriori tests to check necessary conditions for the correctness of the resulting motorcycle graph. In particular, we check (i) for motorcycle traces with a free end[4] and (ii) for motorcycle traces intersecting in the relative interiors of each other. To the best of our knowledge, this is the first competitive motorcycle graph implementation. For this reason, we do not compare our code with other implementations, but content ourselves with a discussion on the performance of Moca.

我们的摩托车代码名为Moca[3]。它使用C++实现，并使用STL来实现常见的数据结构，如队列、红黑树和优先队列。所有的几何计算都基于普通的IEEE 745双精度浮点算术。Moca提供了运行时选项，用于事后测试，以检查所得摩托车图正确性的必要条件。特别是，我们检查（i）具有自由端的摩托车轨迹[4]和（ii）在彼此的相对内部相交的摩托车轨迹。据我们所知，这是第一个具有竞争力的摩托车图实现。因此，我们不将我们的代码与其他实现进行比较，而是满足于讨论Moca的性能。

3.3.3.1 Verification of the stochastic analysis

3.3.3.1 随机分析的验证

We used Moca to collect statistical properties of several datasets, in order to underpin the theoretical results obtained in Section 3.2. We set up three experiments that investigate the dependence of the mean trace length on (i) the number of motorcycles n and (ii) the direction angles δi in Equation (3.10).

我们使用 Moca 收集了几个数据集的统计属性，以便支持第 3.2 节中获得的理论结果。我们设置了三个实验，研究平均轨迹长度对以下因素的依赖性：(i) 摩托车的数量 n 和 (ii) 方程 (3.10) 中的方向角 δi。
For the first two experiments, we created datasets with n random motorcycles by choosing the start points uniformly in $[0, 1]^2$ and the direction angle uniformly from the set $\{0, \delta\}$. Equation (3.10) asserts that the mean trace length L is given by

对于前两个实验，我们创建了包含 n 辆随机摩托车的数据集，方法是在 $[0, 1]^2$ 中均匀选择起点，并从集合 $\{0, \delta\}$ 中均匀选择方向角。方程 (3.10) 断言平均轨迹长度 L 由下式给出

$$E[L] \approx 2p(n-1)|\sin\delta|.$$

(3.11)

In Experiment 1, we created a dataset for each n ∈ {i · 5 000 : 1 ≤ i ≤ 60} and δ ∈ {iπ/12 : 1 ≤ i ≤ 6} and used Moca to determine the mean trace length among each dataset. In the left subfigure of Figure 43, each dot depicts the mean trace length of a dataset, where the resulting values were normalized for illustrative reasons, by dividing them by the factor 2/√n−1. As predicted by Equation 3.11, the plot shows six[5] horizontal lines, where each line corresponds to a particular value of δ. In Experiment 2, we created datasets for n = 10 000 and δ ∈ {iπ/40 : 1 ≤ i ≤ 40}. The reciprocal values of the normalized mean trace lengths are shown in the right subfigure of Figure 43. The normalized mean trace lengths are aligned on the reference curve p| sin δ|, which matches the estimation provided by Equation (3.11).

在实验1中，我们为每个n ∈ {i · 5 000 : 1 ≤ i ≤ 60}和δ ∈ {iπ/12 : 1 ≤ i ≤ 6}创建了一个数据集，并使用Moca确定了每个数据集中的平均轨迹长度。在图43的左侧子图中，每个点表示一个数据集的平均轨迹长度，其中结果值为了说明目的进行了归一化处理，方法是将它们除以因子2/√n−1。正如公式3.11所预测的那样，该图显示了六[5]条水平线，其中每条线对应于δ的特定值。在实验2中，我们为n = 10 000和δ ∈ {iπ/40 : 1 ≤ i ≤ 40}创建了数据集。归一化平均轨迹长度的倒数值显示在图43的右侧子图中。归一化平均轨迹长度与参考曲线p| sin δ|对齐，这与公式（3.11）提供的估计相符。

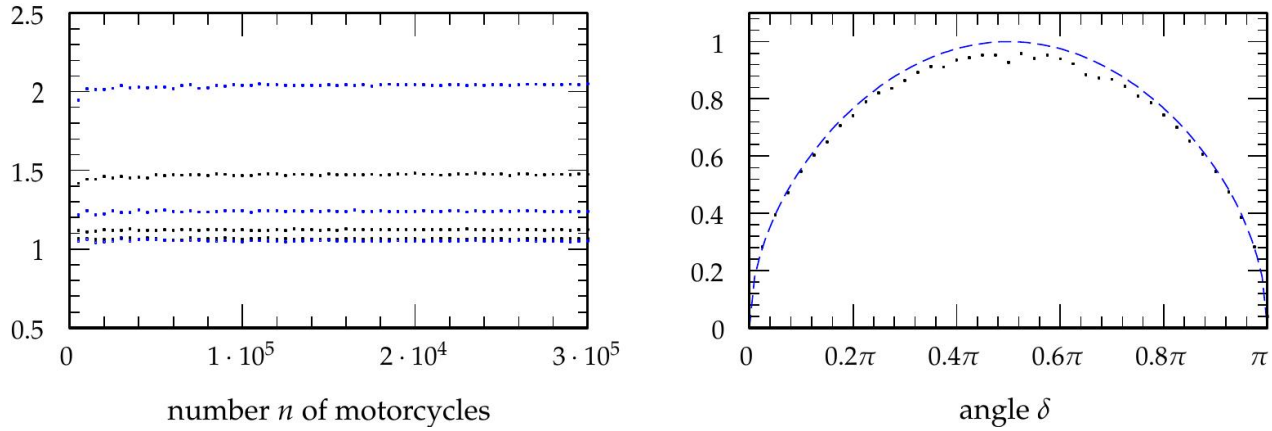


Figure 43: Two experiments illustrating the mean trace length of n motorcycles. A dataset contains motorcycles with uniformly distributed start points and uniformly distributed directions on the set {0, δ}. Left, Experiment 1: every dot depicts the mean trace length for different n and δ. The resulting values are divided by 2/√n−1. Right, Experiment 2: every dot depicts the reciprocal of the mean trace length for a fixed n. The x-axis illustrates δ. The reference curve p| sin δ| is shown in blue.

图 43：两个实验展示了 n 辆摩托车的平均轨迹长度。一个数据集包含具有均匀分布的起点和在集合 {0, δ} 上均匀分布的方向的摩托车。左图，实验 1：每个点表示不同 n 和 δ 的平均轨迹长度。结果值除以 2/√n−1。右图，实验 2：每个点表示固定 n 的平均轨迹长度的倒数。x 轴表示 δ。参考曲线 p| sin δ| 以蓝色显示。

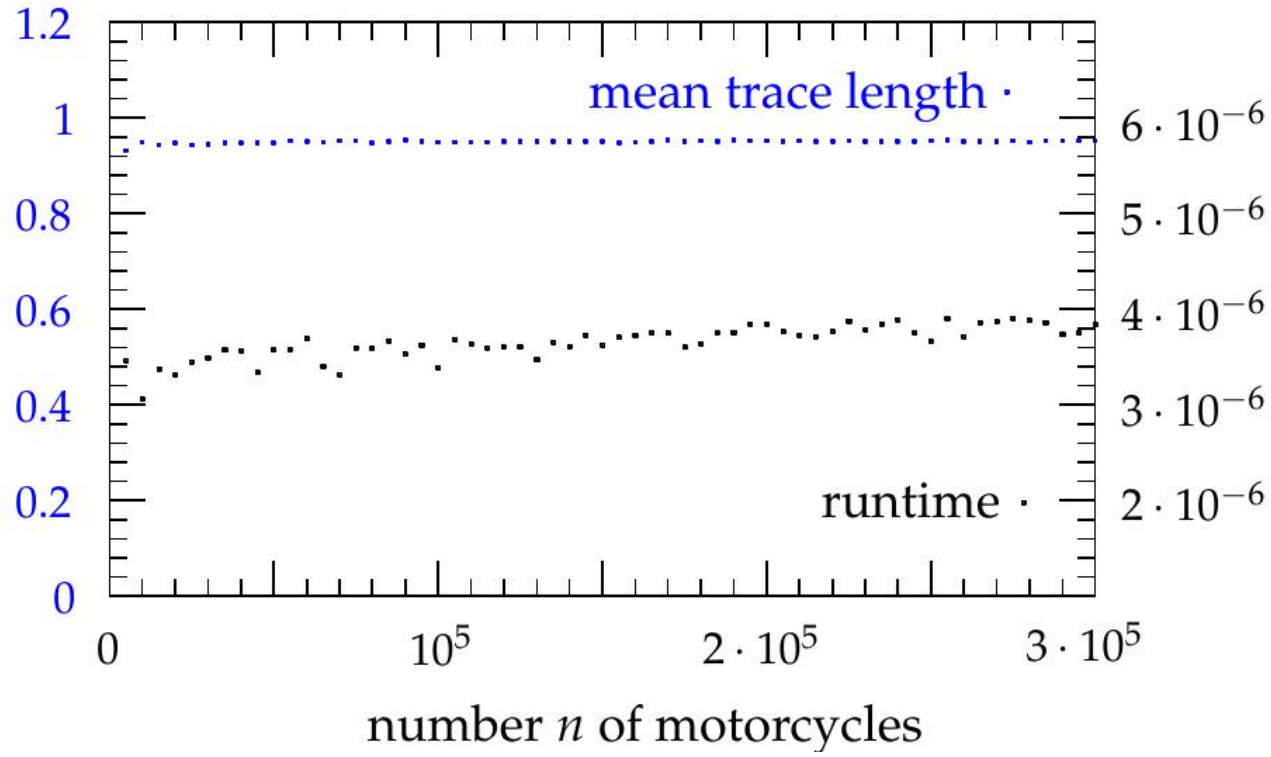


Figure 44: The runtime of Moca and the mean trace length of random dataset containing n motorcycles. The start points are uniformly distributed on $[0, 1]^2$ and the directions are uniformly distributed on [0, 2π). Each blue dot depicts the mean trace length and each black dot the runtime on a single dataset. For illustrative reasons we divided the runtime by n log n and the mean trace length by q n−π1 .

图 44：Moca 的运行时间和包含 n 辆摩托车的随机数据集的平均轨迹长度。起点均匀分布在 $[0, 1]^2$ 上，方向均匀分布在 [0, 2π) 上。每个蓝点表示平均轨迹长度，每个黑点表示单个数据集上的运行时间。出于说明目的，我们将运行时间除以 n log n，将平均轨迹长度除以 q n−π1。

In the third experiment, we considered the mean trace length of datasets where the direction angles are distributed uniformly on [0, 2π). This can be achieved by uniformly distributing the direction angles on {δ1, . . . , δd}, with δi = i 2dπ , and subsequently considering d → ∞. According to Equation (3.10) we obtain:

在第三个实验中，我们考虑了方向角在 [0, 2π) 上均匀分布的数据集的平均轨迹长度。这可以通过在 {δ1, . . . , δd} 上均匀分布方向角来实现，其中 δi = i 2dπ ，随后考虑 d → ∞。根据公式 (3.10) 我们得到：

E

[

L

]

$$\approx \lim_{d \to \infty} d_s \frac{1}{(n-1)} \sum_{d-1} d_{i=1} (d - i$$

)
|
sin
i
2
π
d
|
=
lim
d
→
∞
d
s
1
(
n
−
1
)
.
d

$$
\begin{aligned}
&2\\
&\Sigma\\
&d\\
&-\\
&1\\
&i\\
&=\\
&1\\
&(\\
&1\\
&-\\
&i\\
&d\\
&)\\
&|\\
&\sin\\
&2\\
&\pi\\
&i\\
&d\\
&|\\
&.\\
&1\\
&d\\
&=
\end{aligned}
$$

s

1

(

n

–

1

)

R

1

0

(

1

–

x

)

|

sin

2

π

x

|

d

x

=

r

π

n

−

1

.

(3.12)
We again generated random datasets, where n ranges from 5 000 to 300 000 in steps of 5 000. Figure 44 shows the runtimes and the mean trace lengths on these datasets. The runtime has been divided by n log n and the mean trace lengths have been normalized by the factor √π/n−1. As predicted, the graph shows two horizontal lines. That is, the runtime of Moca is in O(n log n) and the mean trace length is approximately √π/n−1.

我们再次生成了随机数据集，其中n的范围从5 000到300 000，步长为5 000。图44显示了在这些数据集上的运行时间和平均轨迹长度。运行时间已除以n log n，平均轨迹长度已通过因子√π/n−1进行归一化。正如预测的那样，该图显示了两条水平线。也就是说，Moca的运行时间为O(n log n)，平均轨迹长度约为√π/n−1。

3.3.3.2 Runtime statistics on real world data

3.3.3.2 真实世界数据的运行时统计

We performed the following runtime tests on a Linux machine with a 32-bit Kernel. We used an Intel E6700 Core 2 Duo processor, clocked at 2.66 GHz, using 4 GiB of memory. Note that the 32-bit architecture limits the memory footprint of Moca to roughly 3 GiB in user space. For time measurement, we used the C library function getrusage.

我们在一台具有 32 位内核的 Linux 机器上进行了以下运行时测试。我们使用了 Intel E6700 Core 2 Duo 处理器，主频为 2.66 GHz，使用 4 GiB 内存。请注意，32 位架构将 Moca 在用户空间的内存占用限制在约 3 GiB。对于时间测量，我们使用了 C 库函数 getrusage。

Since the development of Moca is motivated by our straight-skeleton implementation Bone, we consider planar straight-line graphs G as input and test Moca by computing the motorcycle graph M(G) induced by G. We ran Moca on more than 22 000 datasets, consisting of synthetic and real-world data. Our real-world datasets include polygonal

crosssections of human organs, GIS maps of roads and river networks, polygonal outlines of fonts, and boundaries of work-pieces for CNC machining or stereo-lithography. The synthetic test data was generated by means of RPG [AH96] and an enhanced version of RPG due to Held. The synthetic data also contains contrived data, like extremely fine approximations of smooth curves, where the vertices are distributed highly irregularly.

由于Moca的开发受到我们直线骨架实现Bone的推动，我们考虑将平面直线图G作为输入，并通过计算由G诱导的摩托车图M(G)来测试Moca。我们在超过22000个数据集上运行了Moca，这些数据集由合成数据和真实世界数据组成。我们的真实世界数据集包括人体器官的多边形横截面、道路和河流网络的GIS地图、字体的多边形轮廓，以及用于CNC加工或立体光刻的工件边界。合成测试数据是通过RPG [AH96]和Held改进的RPG版本生成的。合成数据还包含人为设计的数据，例如对平滑曲线的极其精细的近似，其中顶点分布高度不规则。

Figure 45 (a) illustrates the runtime of Moca on each dataset. The runtimes are given in seconds and were divided by n log n for illustrative reasons. To avoid unreliable timings and other idiosyncrasies of small datasets, we only plot the results of datasets with at least 100 motorcycles. We observe that Moca exhibits a runtime of 2 to 10 · n log n µs for the vast majority of our datasets. About 100 outliers that did not fit into this plot, took up to 2 · n log n ms. A typical dataset of this kind is a sampled ellipse: all motorcycles escape and visit a large number of cells.

图 45 (a) 展示了 Moca 在每个数据集上的运行时间。运行时间以秒为单位给出，并为了便于说明，除以了 n log n。为了避免不可靠的计时和小数据集的其他特性，我们只绘制了至少包含 100 辆摩托车的数据集的结果。我们观察到，对于绝大多数数据集，Moca 的运行时间为 2 到 10 · n log n µs。大约有 100 个不符合此图的异常值，耗时高达 2 · n log n ms。这种类型的典型数据集是一个采样的椭圆：所有摩托车都会逃逸并访问大量的单元格。

The plot in Figure 45 (b) shows the mean trace length, which has been multiplied by √n for a better illustration. We can see that for most datasets in the entire database, the mean

图 45 (b) 中的曲线图显示了平均轨迹长度，为了更好地说明，该长度已乘以 √n。我们可以看到，对于整个数据库中的大多数数据集，平均值

3.3 a simple and practice-minded implementation 93

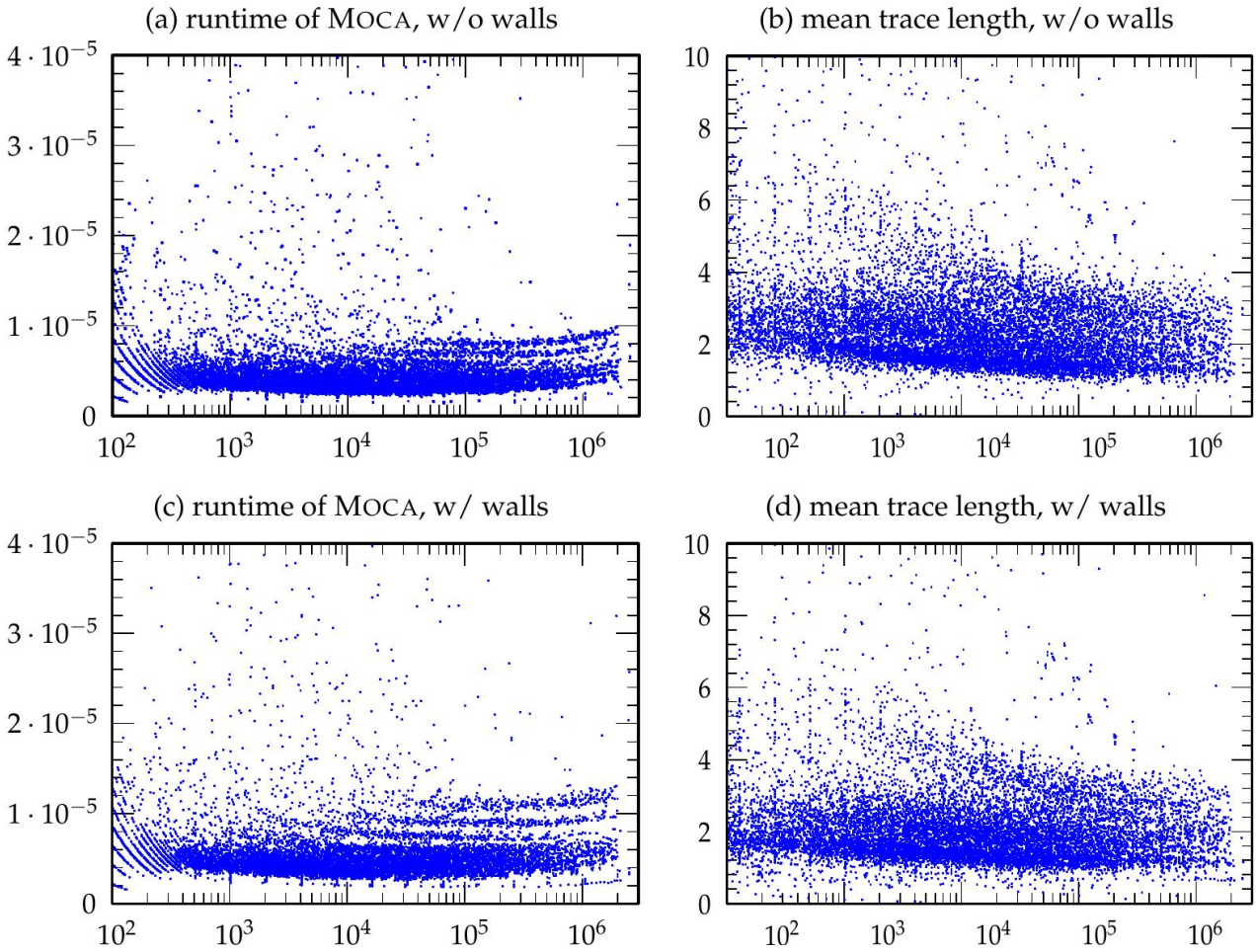


Figure 45: A dot depicts the runtime of Moca resp. the mean trace length of a dataset. The x-axis show the size n of the datasets. (a) Runtime of Moca in seconds, divided by n log n. (b) Mean trace length, multiplied with √n. (c, d) Show the plots analogous to (a, b), but with walls inserted.

图 45：一个点表示 Moca 的运行时，或者一个数据集的平均轨迹长度。x 轴显示数据集的大小 n。(a) Moca 的运行时，单位为秒，除以 n log n。(b) 平均轨迹长度，乘以 √n。(c, d) 显示与 (a, b) 类似的图，但插入了墙。

For the test runs of subfigure (a) and (b), we did not insert the edges of the input graphs G as walls. However, additional tests demonstrate that inserting the walls has only a negligible impact on the runtime of Moca resp. the mean trace length of the resulting motorcycle graphs. We illustrated the corresponding runtimes and mean trace lengths in subfigure (c) and (d).

对于子图 (a) 和 (b) 的测试运行，我们没有将输入图 G 的边作为墙壁插入。然而，额外的测试表明，插入墙壁对 Moca 的运行时间影响甚微。所得摩托车图的平均轨迹长度。我们在子图 (c) 和 (d) 中展示了相应的运行时间和平均轨迹长度。

We further investigated the runtime of Moca on random datasets with non-uniformly distributed start points. For this reason, we generated datasets where the start points resp. the x-coordinates of the start points are distributed Gaussian resp. multi-modal Gaussian. By varying the standard deviations, we are able to successively concentrate the start points at specific regions of the unit square. In Figure 46, we plotted the runtime of Moca and the mean trace lengths on datasets, where (a) the start points are distributed Gaussian at the center of the unit square and (b) only the x-coordinates are distributed Gaussian with mean 0.5. As expected, for a decreasing standard deviation the runtime of Moca increases accordingly. Note that the mean trace length decreases as well, but the impact on the runtime due to the condensed start points dominates the result.

我们进一步研究了Moca在具有非均匀分布起始点的随机数据集上的运行时间。为此，我们生成了数据集，其中起始点分别。起始点的x坐标呈高斯分布。多峰高斯分布。通过改变标准差，我们能够逐步将起始点集中在单位正方形的特定区域。在图46中，我们绘制了Moca的运行时间和数据集上的平均轨迹长度，其中（a）起始点在单位正方形的中心呈高斯分布，（b）只有x坐标呈均值为0.5的高斯分布。正如预期的那样，对于递减的标准差，Moca的运行时间相应增加。请注意，平均轨迹长度也随之减小，但凝聚的起始点对运行时间的影响占据主导地位。

3.3.4 Extending the computation beyond the unit square

3.3.4 将计算扩展到单位正方形之外

The computation of the motorcycle graph outside the unit square can be done very efficiently in terms of worst-case complexities. At first, we compute the motorcycle graph inside the unit square. When a motorcycle reaches the boundary of the unit square it is temporarily stopped. After all motorcycles crashed or have been stopped at the boundary of the unit square, we compute the motorcycle graph outside the unit square. This, however, can be done very efficiently: We interpret the boundary of the growing unit square as a sweep-line. Each motorcycle sits on this growing square and whenever two motorcycles m1, m2 would exchange their positions — i. e., the square reached an intersection point of the tracks of m1 and m2 — then either m1 crashed into m2 or vice versa. The crashed motorcycles are removed from the sweep line and the algorithm continues. The algorithm is correct because the motorcycles are, roughly speaking, moving in the same direction as the wavefront and never against it. Using a balanced binary tree structure on the sweep line allows us to compute the motorcycle graph outside the unit square in O(n log n) time.

在单位正方形外部计算摩托车图，就最坏情况复杂度而言，可以非常高效地完成。首先，我们计算单位正方形内部的摩托车图。当摩托车到达单位正方形的边界时，它会暂时停止。在所有摩托车碰撞或已在单位正方形边界停止后，我们计算单位正方形外部的摩托车图。然而，这可以非常有效地完成：我们将增长的单位正方形的边界解释为扫描线。每辆摩托车都位于这个增长的正方形上，并且每当两辆摩托车m1, m2，m2[/NT0]交换它们的位置时——即，正方形到达了m1和m2的轨迹的交点——那么要么m1撞到了m2，要么反之亦

然。碰撞的摩托车从扫描线中移除，算法继续。该算法是正确的，因为摩托车大致上沿着与波前相同的方向移动，而从不逆着它移动。在扫描线上使用平衡二叉树结构使我们能够在O(n log n)时间内计算单位正方形外部的摩托车图。
This strategy is motivated by the sweep-line algorithm due Eppstein and Erickson [EE99]. Their algorithm computes the motorcycle graph of motorcycles, where the velocities have positive x-coordinates, in O(n log n) time. Note that our approach also works if we use the convex hull of the start points instead of the boundary of the unit square as initial sweep line. In other words, the motorcycle graph outside the convex hull of the start points can be computed in O(n log n) time. However, the computation of the motorcycle graph within the convex hull remains complicated.

此策略的动机源于Eppstein和Erickson提出的扫描线算法[EE99]。他们的算法计算摩托车图，其中摩托车的速度具有正x-坐标，时间复杂度为O(n log n)。请注意，如果我们将起始点的凸包用作初始扫描线，而不是单位正方形的边界，我们的方法也适用。换句话说，起始点凸包外部的摩托车图可以在O(n log n)时间内计算出来。然而，在凸包内计算摩托车图仍然很复杂。

Note that the approach presented above assumes that no motorcycle emerges during the propagation of the sweep line. However, under specific circumstances we can still allow the launch of new motorcycles: (i) the start point needs to coincide with the current position of the wavefront and (ii) the motorcycle needs to drive to the one side of the wavefront that has not yet been swept. Both conditions are fulfilled by our generalization of the motorcycle graph.

请注意，上述方法假设在扫描线传播期间没有摩托车出现。但是，在特定情况下，我们仍然可以允许启动新的摩托车：（i）起点需要与波前的当前位置重合，并且（ii）摩托车需要驶向尚未被扫描的波前的一侧。我们的摩托车图的推广满足了这两个条件。

Moca pursues a simple approach to continue the computation of the motorcycle graph outside the unit square, by extending the 2(√n + 1) grid lines to infinity. We consider the resulting infinite grid cells as part of the hash and continue the simulation of the moving motorcycles on this grid cells. In order to restrict our computations to the unit square, we add four dummy walls that cover the boundary of the unit square. The impact on the performance of Moca, when the boundary walls are removed, directly depends on the number of motorcycles that do not crash within the unit square, which includes the motorcycles that escape.

Moca 采用一种简单的方法来继续计算单位正方形之外的摩托车图，即将 2(√n + 1) 网格线延伸至无穷远。我们将生成的无限网格单元视为哈希的一部分，并继续模拟移动的摩托车在该网格单元上的运动。为了将我们的计算限制在单位正方形内，我们添加了四个虚拟墙，覆盖了单位正方形的边界。移除边界墙后对 Moca 性能的影响，直接取决于未在单位正方形内发生碰撞的摩托车数量，其中包括逃逸的摩托车。

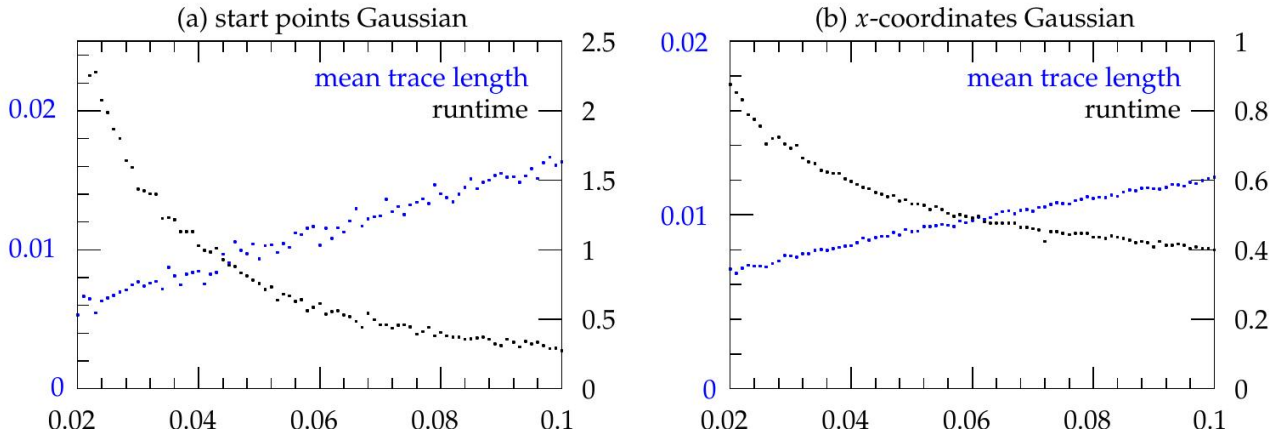


Figure 46: The runtime of Moca and the mean trace length on random datasets, where the direction angles are distributed uniformly on [0, 2π). (a) 2 000 motorcycles, where the start points are distributed Gaussian with mean (0.5, 0.5). (b) 10 000 motorcycles, where the x-coordinates of the start points are distributed Gaussian with mean 0.5 and the y-coordinates are distributed uniformly on [0, 1]. In both subfigures, the x-axis shows the corresponding standard deviation.

图 46：Moca 在随机数据集上的运行时间和平均轨迹长度，其中方向角在 [0, 2π) 上均匀分布。(a) 2 000 辆摩托车，起点呈高斯分布，均值为 (0.5, 0.5)。(b) 10 000 辆摩托车，起点的 x 坐标呈高斯分布，均值为 0.5，y 坐标在 [0, 1] 上均匀分布。在这两个子图中，x 轴显示了相应的标准差。

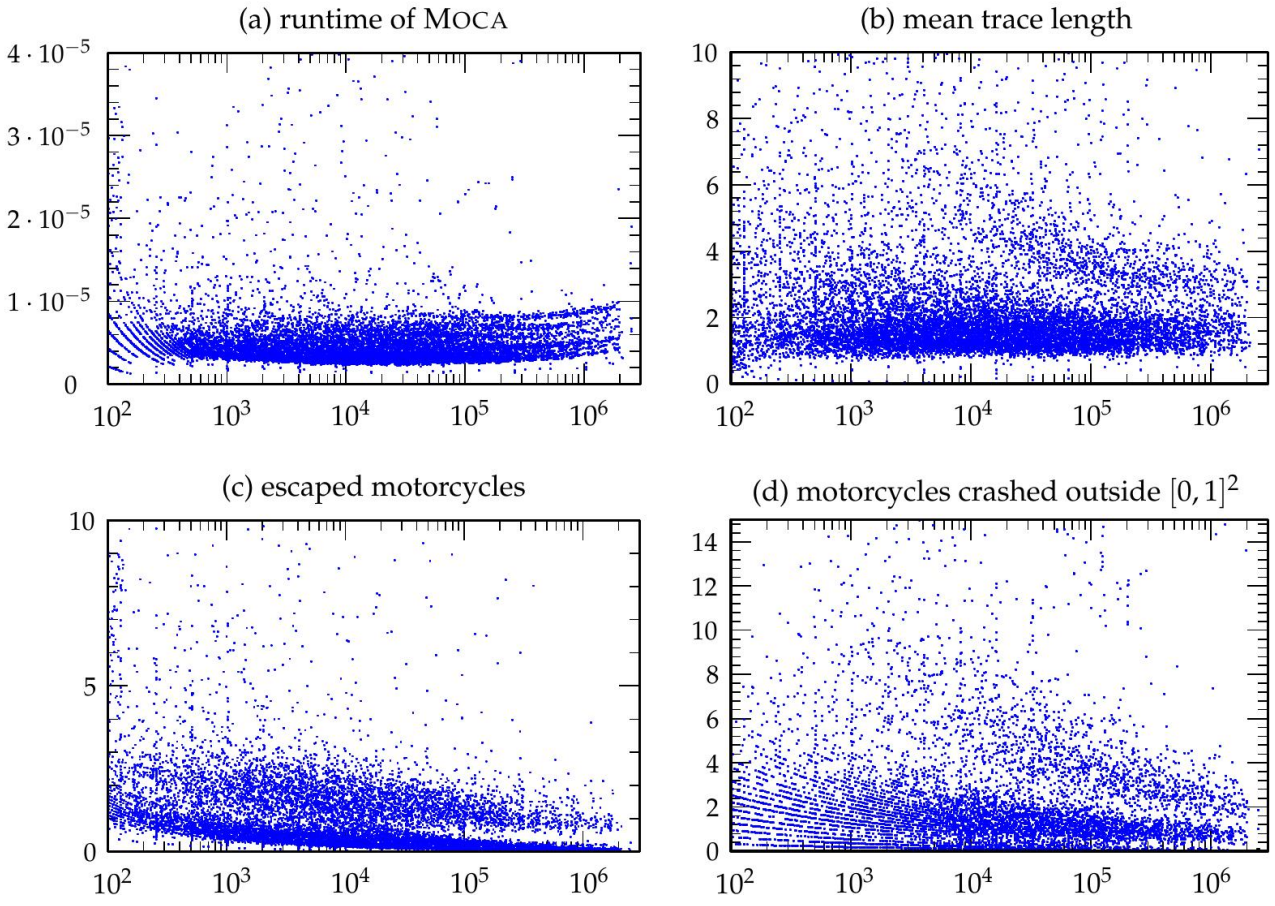


Figure 47: Statistics obtained from Moca when the computation is continued beyond the unit square. The x-axis depicts the number n of motorcycles in the dataset. (a) Runtime, divided by n log n. (b) Mean trace length of the crashed motorcycles, multiplied with √n. (c) Number of escaped motorcycles, divided by √n. (d) Number of motorcycles crashed outside the unit square, divided by √4 n.

图 47：当计算超出单位正方形时，从 Moca 获得的统计数据。x 轴表示数据集中摩托车的数量 n。(a) 运行时间，除以 n log n。(b) 碰撞摩托车的平均轨迹长度，乘以 √n。(c) 逃逸摩托车的数量，除以 √n。(d) 在单位正方形外碰撞的摩托车数量，除以 √4 n。

Figure 47 (a) and (b) show the runtime of Moca and the mean trace length, where the computation is continued beyond the unit square. We observe that there is only a little impact on the runtime of Moca and the same holds for the mean trace length. In subfigure (c) we plotted the number of motorcycles that escaped and subfigure (d) shows the number of motorcycles that crashed outside of the unit square. Note that the remaining motorcycles crashed within the unit square. We observe that the number of motorcycles that escaped is roughly $\Theta(\sqrt{n})$ and the number of motorcycles that crashed outside $[0, 1]^2$ is approximately $\Theta(\sqrt{4}\, n)$. Hence, the vast majority of motorcycles remained within the unit square for most of the datasets in our database. This is the reason why Moca performs well, even though we continue the computation of the motorcycle graph outside the unit square.

图47 (a) 和 (b) 显示了 Moca 的运行时间和平均轨迹长度，其中计算延续到单位正方形之外。我们观察到，对 Moca 的运行时间影响很小，平均轨迹长度也同样如此。在子图 (c) 中，我们绘制了逃逸的摩托车数量，子图 (d) 显示了在单位正方形之外坠毁的摩托车数量。请注意，剩余的摩托车在单位正方形内坠毁。我们观察到，逃逸的摩托车数量大致为 $\Theta(\sqrt{n})$，在 $[0, 1]^2$ 之外坠毁的摩托车数量约为 $\Theta(\sqrt{4}\, n)$。因此，对于我们数据库中的大多数数据集，绝大多数摩托车都保留在单位正方形内。这就是 Moca 表现良好的原因，即使我们继续在单位正方形之外计算摩托车图。

3.4 extracting the motorcycle graph from the straight skeleton

3.4 从直线骨架提取摩托车图

In the introduction of this chapter, we mentioned the close relation between straight skeletons and motorcycle graphs. From a geometric point of view, this relation becomes visible by Theorem 2.26, which states that the motorcycle graph covers the reflex arcs of the straight skeleton. In general, it can be observed that the gap between the reflex arcs and the motorcycle traces decreases as the motorcycles resp. the reflex wavefront vertices move faster. From this observation, the question arises whether we can always approximate the motorcycle graph using the straight skeleton. In other words, can we compute the motorcycle graph using a straight-skeleton algorithm?

在本章的引言中，我们提到了直线骨架和摩托车图之间的密切关系。从几何的角度来看，这种关系通过定理2.26变得明显，该定理指出摩托车图覆盖了直线骨架的反射弧。一般来说，可以观察到，随着摩托车（或反射波前顶点）移动得更快，反射弧和摩托车轨迹之间

的差距会减小。从这个观察结果来看，出现了一个问题：我们是否总是可以使用直线骨架来近似摩托车图。换句话说，我们是否可以使用直线骨架算法来计算摩托车图？换句话说，我们是否可以使用直线骨架算法来计算摩托车图？
We think that this question is interesting due to following reasons. Firstly, since motorcycle graphs play an important role in the computation of straight skeletons, it appears to be important to deepen the insight into the geometric relation between motorcycle graphs and straight skeletons. Secondly, if we manage to efficiently reduce the construction problem of motorcycle graphs to straight skeletons then we are able to describe the complexity of one problem in terms of the other problem. In fact, in Section 3.4.3, we present a proof for the P-completeness of straight skeletons that is based on the results of the subsequent sections and the P-completeness of motorcycle graphs due to Eppstein and Erickson [EE99].

我们认为这个问题很有趣，原因如下。首先，由于摩托车图在直线骨架的计算中起着重要作用，因此加深对摩托车图和直线骨架之间几何关系的理解显得非常重要。其次，如果我们能够有效地将摩托车图的构造问题简化为直线骨架的构造问题，那么我们就能够用另一个问题的复杂度来描述一个问题的复杂度。事实上，在第3.4.3节中，我们给出了直线骨架的P-完全性的证明，该证明基于后续章节的结果以及Eppstein和Erickson [EE99]提出的摩托车图的P-完全性。

3.4.1 Approximating the motorcycle graph by the straight skeleton

3.4.1 用直线骨架逼近摩托车图

We assume that n motorcycles m1, . . . , mn are given, where each motorcycle mi has a start point pi and a speed vector vi. All motorcycles start at the same time. In particular, no motorcycles emerge later on. Can we find an appropriate planar straight-line graph G such that the straight skeleton S (G), resp. a proper subset of S (G), approximates M(m1, . . . , mn) up to a given tolerance?

我们假设给定了n辆摩托车m1, . . . , mn，其中每辆摩托车mi都有一个起点pi和一个速度向量vi。所有摩托车同时启动。特别地，没有摩托车在之后出现。我们能否找到一个合适的平面直线图G，使得直线骨架S(G)，或其适当子集，在给定的容差范围内逼近M(m1, . . . , mn)？
It follows from Theorem 2.26 that if we construct G such that at each pi a reflex wavefront vertex starts moving along the track of mi with velocity vi then the reflex arcs of S (G) that belong to these wavefront vertices approximate the traces of M(m1, . . . , mn) up to some gap. The simplest way in order to obtain such reflex wavefront vertices is to place isosceles triangles Δi at each pi such that the trace of mi is bisecting the exterior angle of Δi. By setting the angle of Δi at pi accordingly, we can adapt the speed of the corresponding wavefront vertex.

由定理 2.26 可知，如果我们构造 G，使得在每个 pi 处，一个反射波前顶点开始沿着 mi 的轨迹以速度 vi 移动，那么属于这些波前顶点的 S (G) 的反射弧将逼近 M(m1, . . . , mn) 的轨迹，直至某个间隙。获得这种反射波前顶点的最简单方法是在每个 pi 处放置等腰三角形

∆i，使得 mi 的轨迹平分 ∆i 的外角。通过相应地设置 ∆i 在 pi 处的角度，我们可以调整相应波前顶点的速度。

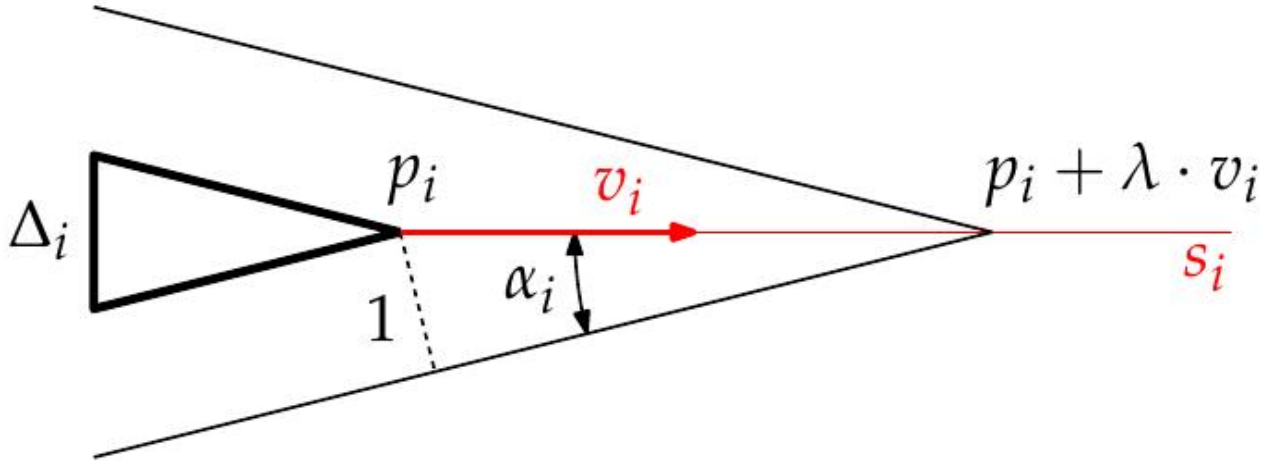


Figure 48: The isosceles triangle ∆i is placed on the vertex pi. The interior angle of ∆i at pi is 2αi and λ|vi| = 1/sin αi.

图 48：等腰三角形 ∆i 放置在顶点 pi 上。∆i 在 pi 处的内角为 2αi，且 λ|vi| = 1/sin αi。

As already mentioned, we observe that the faster m moves the better is its trace approximated by the reflex straight-skeleton arc that is covered by m. But since the speed of m1, . . . , mn is part of the input to our problem setting, we cannot change the speeds of the motorcycles. Moreover, a wavefront vertex has always a speed of at least 1. If a motorcycle has a speed less than 1 then we cannot construct an according ∆i, as mentioned above. However, it easy to see that if we multiply each speed vector vi by the same constant λ > 0 then the motorcycle graph remains the same. Hence, the idea is to place at each pi an isosceles triangle ∆i, where the interior angle at pi equals 2αi and αi is given by

正如已经提到的，我们观察到m移动得越快，其轨迹就越能被m覆盖的反射直线骨架弧近似。但是由于m1, . . . , mn的速度是我们问题设置的输入的一部分，我们不能改变摩托车的速度。此外，波前顶点的速度始终至少为1。如果摩托车的速度小于1，那么我们无法构建相应的∆i，如上所述。然而，很容易看出，如果我们将每个速度向量vi乘以相同的常数λ > 0，那么摩托车图保持不变。因此，我们的想法是在每个pi处放置一个等腰三角形∆i，其中pi处的内角等于2αi，并且αi由下式给出

$$\alpha_i := \arccos\left(\frac{\lambda \cdot |v_i| - 1}{\lambda \cdot |v_i| + 1}\right)$$

csin

1

λ

|

v

i

|

.

(3.13)

The constant λ is chosen large enough such that λ|vi| > 1 for all 1 ≤ i ≤ n, see Figure 48. The size of ∆i will be specified later. We denote by ui the reflex wavefront vertex emanated at pi. Note that ui has λ-times the speed of mi. Furthermore, note that each triangle ∆i is emanating two additional motorcycles at the other two corners. This leads us to a refined version of our initial question: Can we find λ large enough such that the reflex arcs of S (G) that are emanated from pi, approximate M(m1, . . . , mn) up to a given tolerance?

选择足够大的常数 λ，使得对于所有 1 ≤ i ≤ n，有 λ|vi| > 1，参见图 48。∆i 的大小将在稍后指定。我们用 ui 表示从 pi 发出的反射波前顶点。注意，ui 的速度是 mi 的 λ 倍。此外，请注意，每个三角形 ∆i 都在另外两个角上发出两个额外的摩托车。这引导我们对最初的问题进行改进：我们能否找到足够大的 λ，使得从 pi 发出的 S (G) 的反射弧，在给定的容差范围内逼近 M(m1, . . . , mn)？

We denote the trace of each motorcycle mi by si. Recall that Dμ denotes the disk with radius μ and the center at the origin. Since all traces are closed sets, there exists μ > 0 such that the Minkowski sums si + Dμ and sj + Dμ are disjoint for all 1 ≤ i, j ≤ n, where si and sj are disjoint. For example, let μ be a third of the pairwise infimum distance among all disjoint traces. Note that two traces si, sj are intersecting if and only if mi crashed into mj or vice versa. Further, we choose μ small enough such that pi + Dμ is disjoint with sj + Dμ for all 1 ≤ i, j ≤ n, where pi is disjoint to sj. We denote by G the planar straight-line graph that consists of the triangles ∆i, as described above, where the lengths of the arms that are incident to pi are set to μ/2.

我们用si表示每辆摩托车mi的轨迹。回想一下，Dμ表示以原点为中心、半径为μ的圆盘。由于所有轨迹都是闭集，因此存在μ > 0，使得对于所有1 ≤ i, j ≤ n，Minkowski和si + Dμ和sj + Dμ是不相交的，其中si和sj是不相交的。例如，设μ是所有不相交轨迹之间成对下确界距离的三分之一。请注意，两条轨迹si, sj相交当且仅当mi撞到mj或反之亦然。此外，我们选

择足够小的μ，使得对于所有1 ≤ i, j ≤ n，pi + Dμ与sj + Dμ是不相交的，其中pi与sj是不相交的。我们用G表示平面直线图，它由如上所述的三角形∆i组成，其中与pi相连的边的长度设置为μ/2。
Lemma 3.4. The wavefronts of ∆i stay within si + Dμ until time μ/4 if λ ≥ |v2i| .

引理 3.4. 如果 λ ≥ |v2i|，则 ∆i 的波前保持在 si + Dμ 内，直到时间 μ/4。

Proof. From λ ≥ 2/|vi| follows that 2αi is at most 60◦ . Hence, the other two angles of ∆i are at least 60◦ and the two additional motorcycles at ∆i have a speed of at most 2. Since the start points of those motorcycles have a distance of μ/2 from pi and since they drive at most a distance of μ/2 in time μ/4, they stay within pi + Dμ.

证明。从λ ≥ 2/|vi|可知，2αi最多为60◦。因此，∆i的另外两个角至少为60◦，并且∆i处的另外两辆摩托车的速度最多为2。由于这些摩托车的起点与pi的距离为μ/2，并且由于它们在时间μ/4内最多行驶μ/2的距离，因此它们停留在pi + Dμ内。
We denote by −→si the ray that starts at pi in direction vi and define

我们用−→si表示从pi出发，沿vi方向的射线，并定义

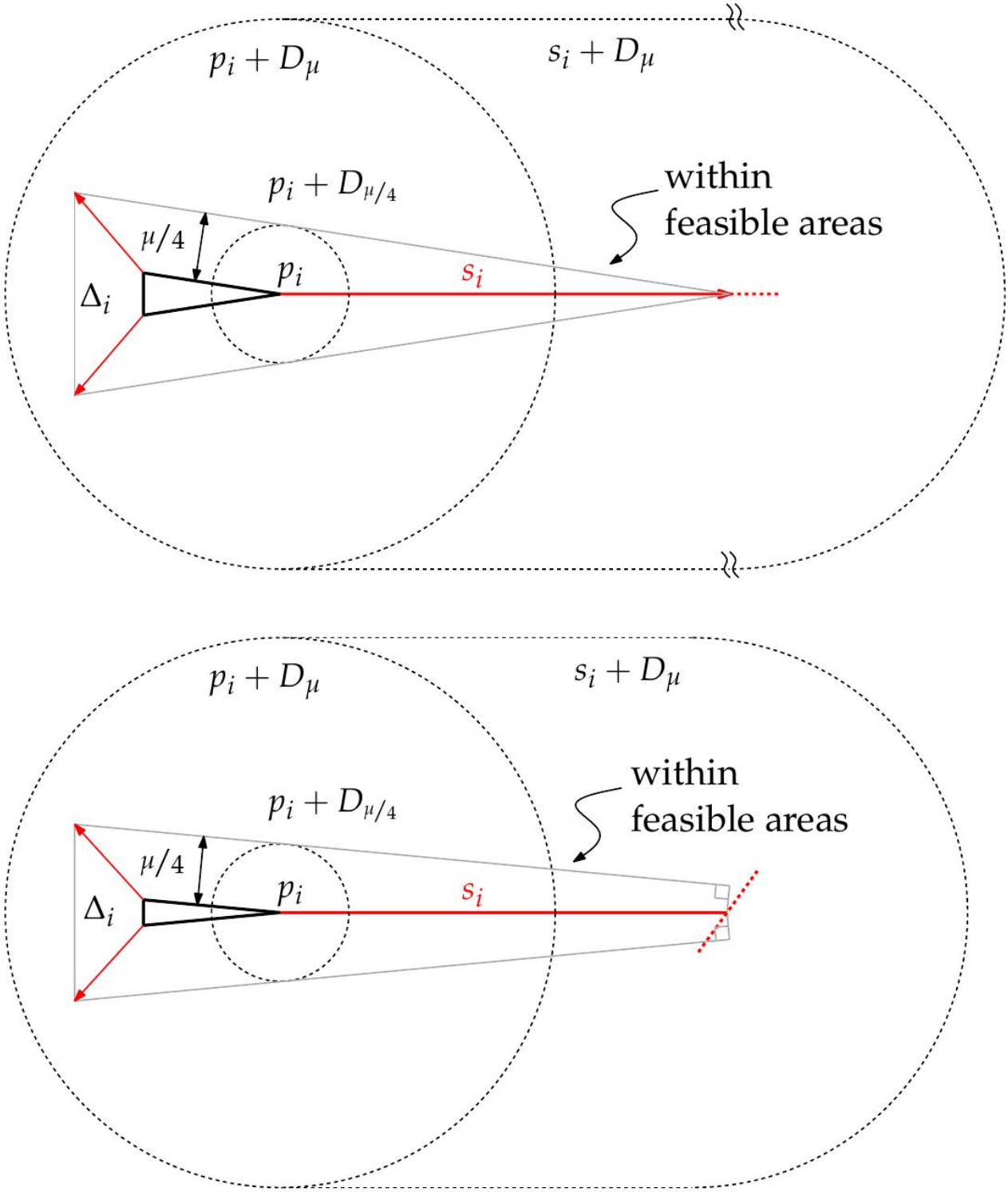


Figure 49: The wavefronts of ∆i are bounded to si + Dμ until time μ/4. Top: the motorcycle mi did not crash until time μ/4. Bottom: the motorcycle mi did crash until time μ/4.

图 49：∆i 的波前被限制在 si + Dμ 内，直到时间 μ/4。上图：摩托车 mi 在时间 μ/4 之前没有发生碰撞。下图：摩托车 mi 在时间 μ/4 之前发生了碰撞。

Next, we consider a wavefront edge e that is emanated from ∆i and propagating to the exterior of ∆i. Let us recall the slabs of the lower envelope from Section 2.4.3. We call the slab that belongs to e and which is vertically projected onto the plane the feasible area of e. The face f (e) is contained in the feasible area of e. We have to proof that e(t) stays within si + Dμ until time μ/4. First, we restrict the feasible area of e to those points that

have an orthogonal distance of at most μ/4 to e(0), see Figure 49. We distinguish two cases: the motorcycle mi did crash or did not crash until time μ/4. In both cases the corner points of the restricted feasible areas are contained within si + Dμ, the restricted feasible areas are convex and si + Dμ is convex. Hence, the wavefronts of ∆i until time μ/4 are contained within si + Dμ.

接下来，我们考虑一个从∆i发出并向∆i外部传播的波前边缘e。让我们回顾一下第2.4.3节中下包络面的板。我们将属于e并垂直投影到平面上的板称为e的可行区域。面f (e)包含在e的可行区域内。我们必须证明e(t)在时间μ/4之前保持在si + Dμ内。首先，我们将e的可行区域限制为到e(0)的正交距离最多为μ/4的点，参见图49。我们区分两种情况：摩托车mi在时间μ/4之前发生碰撞或未发生碰撞。在这两种情况下，受限可行区域的角点都包含在si + Dμ内，受限可行区域是凸的，并且si + Dμ是凸的。因此，∆i在时间μ/4之前的波前包含在si + Dμ内。

$$L := \max_{1 \le i, j \le n} d(p_i, \vec{\ })$$

S

i

∩

−

→

S

j

)

.

(3.14)

Note that we may only consider indices i, j for which −→si ∩ −→sj is not empty. If no such indices i, j exist then we set L to zero. Further, let us denote by ϕi,j ∈ [0, π] the non-oriented angle spanned by vi and vj, with ϕi,j = ϕj,i. Next we define

请注意，我们可能只考虑使得−→si ∩ −→sj非空的索引i, j。如果不存在这样的索引i, j，那么我们将L设为零。更进一步，我们用ϕi,j ∈ [0, π]表示由vi和vj所张成的非定向角，其中ϕi,j = ϕj,i。接下来我们定义

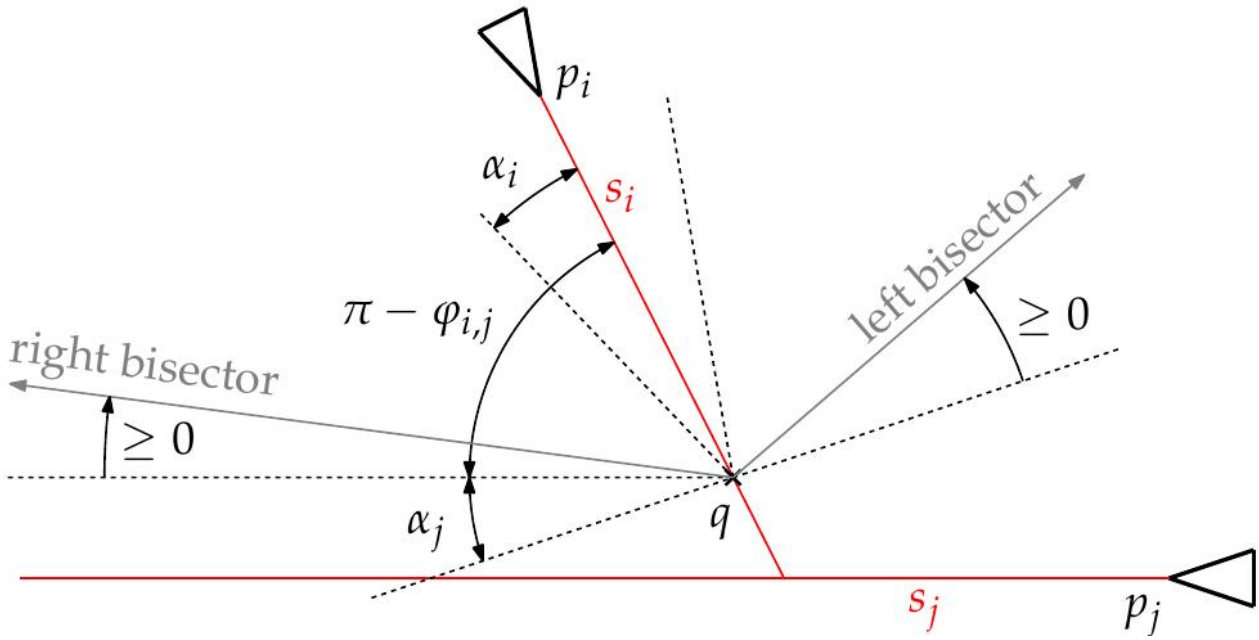


Figure 50: The wavefronts of ∆i do not cause a crash of the reflex wavefront vertex from pj.

图 50：∆i 的波前不会导致来自 pj 的反射波前顶点崩溃。

$$
\Phi := \min_{1 \le i < j \le n} \left\{ \phi_{i,j}, \pi - \phi_{i,j} \right\}
$$

.

(3.15)

If the corresponding set it is empty — i.e. if all motorcycles drive on parallel tracks — then we set Φ := π/2.

如果对应的集合是空的——即如果所有摩托车都在平行轨道上行驶——那么我们设置Φ := π/2。

Lemma 3.5. Let mi denote a motorcycle that crashes into the motorcycle mj. The wavefronts of ∆i do not cause a split event for the reflex wavefront vertex uj until time µ/4 if λ ≥ 2/mink |vk| sin Φ.

引理 3.5。设 mi 表示撞向摩托车 mj 的摩托车。如果 λ ≥ 2/mink |vk| sin Φ，则 ∆i 的波前在时间 µ/4 之前不会导致反射波前顶点 uj 发生分裂事件。

Proof. Since λ ≥ |vk| sin2 Φ holds for any 1 ≤ k ≤ n, it follows that

证明。由于对于任何 1 ≤ k ≤ n，λ ≥ |vk| sin2 Φ 成立，因此可得

$$\sin\frac{\alpha_k}{2} \leq \frac{1}{|v_k|\lambda} \leq \frac{1}{2\sin$$

Φ

≤

sin

Φ

2

,

because sin is concave on [0, π]. By further noting that sin is monotone on [0, π/2] we see that

因为正弦函数在[0, π]上是凹函数。通过进一步注意到正弦函数在[0, π/2]上是单调的，我们看到

α

k

≤

Φ

2

∀

1

≤

k

≤

n

.

The motorcycle mj does not also crash into mi, since two motorcycles do not crash simultaneously into each other by assumption. If si and sj are collinear then the assertion is either trivial or excluded by assumption. Without loss of generality, we may assume that

si is right of $\overrightarrow{-s_j}$, see Figure 50. We denote by q the endpoint of the reflex straight-skeleton arc that is incident to pi. Let us consider the left (resp. right) bisector between the left (resp. right) arm of mi and the right arm of mj, starting from q.

摩托车mj也不会撞到mi，因为根据假设，两辆摩托车不会同时互相碰撞。如果si和sj共线，则该断言要么是微不足道的，要么被假设排除。不失一般性，我们可以假设si位于$\overrightarrow{-s_j}$的右侧，参见图50。我们用q表示与pi相关的反射直线骨架弧的端点。让我们考虑从q开始的mi的左（右）臂和mj的右臂之间的左（右）平分线。
By Lemma 3.4 it suffices to show that the two arms of mi do not lead to a split event with uj until time $\mu/4$: The two additional motorcycles from Δ_i stay within $p_i + D_\mu$. Hence, we only have to consider the arms of mi.

根据引理 3.4，只需证明 mi 的两个分支在时间 $\mu/4$: 之前不会导致与 uj 的分裂事件：来自 Δ_i 的两辆额外摩托车停留在 $p_i + D_\mu$ 内。因此，我们只需要考虑 mi 的分支。
We conclude the proof by showing that none of both bisectors intersects $\overrightarrow{-s_j}$. Let us consider the right bisector. Recall that $\alpha_i, \alpha_j \leq \Phi/2$ and that $\pi - \phi_{i,j} \geq \Phi$. In the extremal case, where equality is attained for all three inequalities, the right bisector is just parallel to sj, but strictly right of $\overrightarrow{-s_i}$. In all other cases the bisector rotates clockwise at q such that our assertion is true in general. Analogous arguments hold for the left bisector. Summarizing, the reflex wavefront vertex uj does not lead to a split event with the wavefronts of Δ_i until time $\mu/4$.

我们通过证明两个角平分线均不与 $\overrightarrow{-s_j}$ 相交来结束证明。让我们考虑垂直平分线。回顾 $\alpha_i, \alpha_j \leq \Phi/2$ 且 $\pi - \phi_{i,j} \geq \Phi$。在极端情况下，当所有三个不等式都达到等号时，垂直平分线恰好平行于 sj，但严格位于 $\overrightarrow{-s_i}$ 的右侧。在所有其他情况下，角平分线在 q 处顺时针旋转，因此我们的断言在一般情况下成立。类似的论证适用于左侧角平分线。总而言之，反射波前顶点 uj 在时间 $\mu/4$ 之前不会导致与 Δ_i 的波前发生分裂事件。

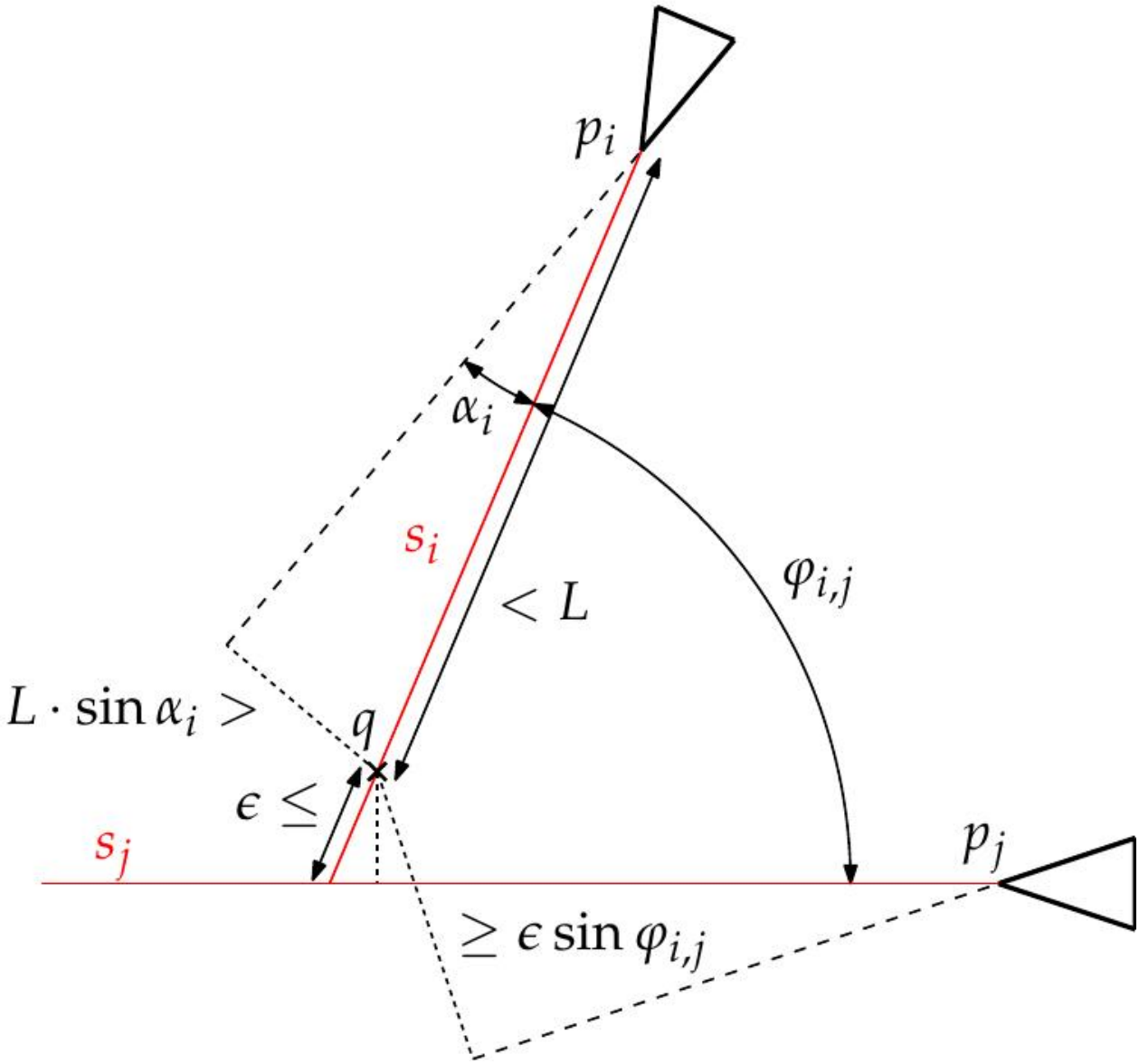


Figure 51: The point q is reached earlier by the left arm of mi than by the right arm of mj.

图 51：mi 的左臂比 mj 的右臂更早到达点 q。

Lemma 3.6. Let mi denote a motorcycle crashing into the motorcycle mj. For any ϵ > 0 and

引理 3.6。设 mi 表示摩托车撞击摩托车 mj。对于任何 ε > 0 且

$$\lambda \geq \frac{1}{\min_k}$$

|

v

k

|

.

sin

Φ

.

max

□

2,

L

min

{

μ

/

4

,

∈

}

□

,

the trace si is covered up to a gap size ϵ by the reflex arc traced out by ui.

轨迹si被ui描绘出的反射弧覆盖，直至间隙大小达到ϵ。

Proof. We will prove the following: any point q on si, whose distance to the endpoint of si is at least ϵ, is reached by ui no later than at time μ/4, see Figure 51. We first show that until time μ/4 the reflex wavefront vertex ui may only cause a split event with the wavefronts of ∆j. By Lemma 3.4, we know that until time μ/4, the wavefronts of a triangle ∆k could only cause a split event with ui if sk and si intersect. Hence, mk crashed against si. However, by Lemma 3.5 it follows that ui does not lead to a split event with the wavefronts from ∆k.

证明。我们将证明以下内容：si上的任何点q，其到si端点的距离至少为ϵ，都将在不晚于μ/4的时间被ui到达，见图51。我们首先证明，在时间μ/4之前，反射波前顶点ui可能只会与∆j的波前发生分裂事件。根据引理3.4，我们知道在时间μ/4之前，三角形∆k的波前只有在sk和si相交时，才可能与ui发生分裂事件。因此，mk撞击了si。然而，根据引理3.5，可以得出ui不会导致与来自∆k的波前发生分裂事件。

W.l.o.g., we may assume that si lies on the right of $\overrightarrow{-}sj$. In order to show that ui reaches q until time μ/4, it suffices to prove that q has a smaller orthogonal distance to the left arm of mi than to the right arm of mj and that the orthogonal distance of q to the left arm of mi is at most μ/4.

不失一般性，我们可以假设si位于$\overrightarrow{-}sj$的右侧。为了证明ui在时间μ/4之前到达q，只需证明q到mi左臂的正交距离小于到mj右臂的正交距离，并且q到mi左臂的正交距离最多为μ/4。

The orthogonal distance of q to the left arm of mi is at most $L \cdot \sin \alpha_i$. The orthogonal distance of q to the right arm of mj is at least the orthogonal distance of q to sj. However, this distance is at least $\epsilon \cdot \sin \phi_{i,j}$. Summarizing, our assertion holds if

正交距离 q 到 mi 的左臂至多为 $L \cdot \sin \alpha_i$。正交距离 q 到 mj 的右臂至少为 q 到 sj 的正交距离。然而，这个距离至少为 $\epsilon \cdot \sin \phi_{i,j}$。总而言之，如果以下条件成立，我们的断言就成立

$$L \cdot \sin \alpha_i \leq \min \{ \mu$$

/

4

,

∈

sin

ϕ

i

,

j

}

,

and thus

因此

λ

≥

L

|

v

i

|

.

min

{

μ

/

4

,

∈

sin

ϕ

i

,

j

}

.
Our choice for λ fulfills this condition. The case where sj and si are collinear such that mi crashes at pj is similar. The wavefront of ∆j reaches q no later than at time ϵ/2 and vi reaches q in at most λ|Lvi| time.

我们选择的λ满足这个条件。当sj和si共线，使得mi在pj处崩溃的情况是类似的。∆j的波前到达q的时间不晚于ϵ/2，并且vi到达q的时间最多为λ|Lvi|。

Let us denote by Sλ∗(m1, . . . , mn) ⊂ S (G) the union of the reflex straight-skeleton arcs that are traced out by u1, . . . , un, where G is given as described above. Then we get the following corollary of Lemma 3.6:

令 Sλ∗(m1, . . . , mn) ⊂ S (G) 表示由 u1, . . . , un 描绘出的反射直线骨架弧的并集，其中 G 如上所述。那么我们得到引理 3.6 的以下推论：

Corollary 3.7.

推论 3.7.

lim

λ

→

∞

S

∗

λ

(

m

1

,

.

.

.

,

m

n

)

=

M

(

m

1

,

.

.

.

,

m

n

)

This corollary also includes that a point q on a motorcycle trace si of a motorcycle mi that never crashed, is covered by an arc of Sλ∗(m1, . . . , mn) for large enough λ. However, this is easy to see by applying Lemma 3.4 and Lemma 3.5, and by finally finding λ large enough such that the point q is reached by ui until time µ/4.

这个推论还包括，永不崩溃的摩托车mi的摩托车轨迹si上的点q，对于足够大的λ，被Sλ∗(m1, . . . , mn)的弧覆盖。然而，通过应用引理3.4和引理3.5，并最终找到足够大的λ，使得点q在时间μ/4之前被ui到达，这很容易看出。
3.4.2 Computing the motorcycle graph

3.4.2 计算摩托车图

In order to reduce the construction problem of motorcycle graphs to straight skeletons, we have to cope with the remaining gaps between the motorcycle traces si and the reflex arcs traced out by ui, which exist for arbitrary large λ. Hence, the question remains whether mi actually crashed into mj (or vice versa), even if the gap between two reflex arcs that are traced out by ui and uj, is very small.

为了将摩托车图的构造问题简化为直线骨架，我们必须处理摩托车轨迹si与由ui描绘出的反射弧之间剩余的间隙，这些间隙对于任意大的λ都存在。因此，问题仍然存在：mi是否真的撞到了mj（或者反之），即使由ui和uj描绘出的两个反射弧之间的间隙非常小。

In order to decide whether the motorcycle mi escapes or whether it crashes into a trace sj, we determine λ large enough such that the following conditions are fulfilled:

为了确定摩托车mi是逃逸还是撞到轨迹sj，我们确定足够大的λ，以满足以下条件：

• If mi crashes into a trace sj then the reflex wavefront vertex ui leads to a split event until the time μ/4 and the reflex arc that is traced out by ui has an endpoint in a straightskeleton face of an edge of ∆j. (The vertex ui causes a split event with the right arm of mj if si is right of −→sj and the left arm if si is left of −→sj .)

• 如果mi撞击到轨迹sj，则反射波前顶点ui会导致分裂事件，直到时间μ/4，并且由ui描绘出的反射弧在∆j的边的直线骨架面上有一个端点。（如果si在−→sj的右侧，则顶点ui会导致与mj的右臂的分裂事件；如果si在−→sj的左侧，则与左臂分裂。）

• If mi escapes then the reflex wavefront vertex ui did not lead to a split event until the time μ/4.

• 如果mi逃逸，则反射波前顶点ui在时间μ/4之前没有导致分裂事件。

The following lemma states that such λ exists and provides a sufficient bound.

以下引理表明存在这样的λ，并提供了一个充分的界限。

Lemma 3.8. Consider S (G) with

引理 3.8. 考虑具有以下性质的 S (G)：

$$\lambda$$

$$\geq$$

max
n
2,
8
L
μ
o
min
k
|
v
k
|
·
sin
Φ
.
Then mi crashes into sj if and only if ui leads to a split event with the wavefront emanated by ∆j until time μ/4. In particular, mi escapes if and only if ui does not lead to a split event until time μ/4.

那么，mi撞击sj当且仅当ui导致一个分裂事件，该事件具有由∆j发出的波前，直到时间μ/4。特别地，mi逃逸当且仅当ui在时间μ/4之前没有导致分裂事件。
Proof. We distinguish two cases. First, suppose that the motorcycle mi crashed into the trace sj, see Figure 52. We may assume without loss of generality that si is right of −→sj . First we note that by our choice of λ we may apply Lemma 3.4. We denote by p the intersection si ∩ sj. Further, we set ϵ := μ/8, which allows us to apply Lemma 3.6, because

证明。我们区分两种情况。首先，假设摩托车 mi 撞到了轨迹 sj，见图 52。我们可以不失一般性地假设 si 在 −→sj 的右侧。首先，我们注意到，根据我们对 λ 的选择，我们可以应用引理 3.4。我们用 p 表示交点 si ∩ sj。此外，我们设置 ϵ := μ/8，这使我们能够应用引理 3.6，因为

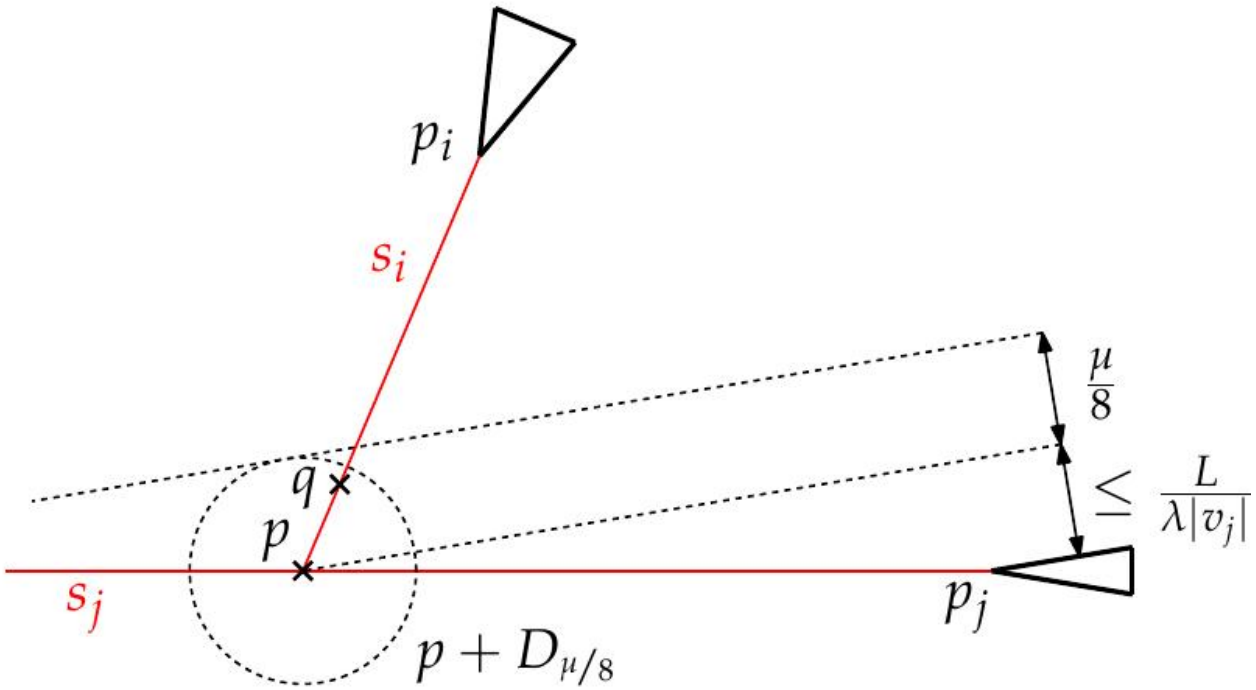


Figure 52: The reflex wavefront vertex ui causes a split event until time μ/4.

图 52：反射波前顶点 ui 导致分裂事件，直至时间 μ/4。

$$\max_{2,} \frac{8L}{\mu} \geq \max_{2,} L$$

min

{

μ

/

4

,

∈

}

□

.

Thus, the endpoint q of the reflex arc that is traced out by ui, has a distance of at most μ/8 to p. On the other hand, uj reaches p at time L/λ|vj| at the latest. We conclude that the right arm of mj reaches q at time L/λ|vj| + μ/8 at the latest, which is bounded from above by

因此，由ui描绘出的反射弧的端点q到p的距离最多为μ/8。另一方面，uj最迟在时间L/λ|vj|到达p。我们得出结论，mj的右臂最迟在时间L/λ|vj| + μ/8到达q，其上限为

L

.

μ

.

min

k

|

v

k

|

.

sin

Φ

8

.

L

.

|

v

j

|

+

μ

8

≤

μ

4

by our choice of λ. Summarizing, the point q is swept by the wavefront of the right arm of mj and is reached by ui until μ/4 time. Hence, ui must have caused a split event until the requested time by crashing into the wavefront of the right arm of mj.

由我们选择的λ。总结一下，点q被mj右臂的波前扫过，并且在μ/4时间之前被ui到达。因此，ui必定在所请求的时间之前，通过撞击mj右臂的波前而导致了一次分裂事件。

For the second case assume that mi escapes. Lemma 3.4 and Lemma 3.5 imply that ui does not lead to a split event until time μ/4.

对于第二种情况，假设 mi 逃逸。引理 3.4 和引理 3.5 表明，ui 在时间 μ/4 之前不会导致分裂事件。

The previous lemma enables us to compute the motorcycle graph by employing a straight skeleton algorithm. However, in order to apply the lemma, we need to compute appropriate values for L, Φ, μ in order to determine a sufficiently large λ. While L and Φ are already given independent of M(m1, . . . , mn), the following lemma gives a formula for μ, for which the actual motorcycle graph is not needed to be known. (In the following lemma we take d(−→si , ∅) to be infinity.)

先前的引理使我们能够通过使用直线骨架算法来计算摩托车图。然而，为了应用该引理，我们需要计算 L、Φ、μ 的适当值，以便确定一个足够大的 λ。虽然 L 和 Φ 已经给出，与 M(m1, . . . , mn) 无关，但以下引理给出了 μ 的公式，而无需知道实际的摩托车图。（在以下引理中，我们取 d(−→si , ∅) 为无穷大。）

Lemma 3.9. For any two disjoint motorcycle traces si and sj the Minkowski sums si + Dμ and sj + Dμ are disjoint for

引理 3.9. 对于任意两个不相交的摩托车轨迹 si 和 sj，Minkowski 和 si + Dμ 和 sj + Dμ 是不相交的，因为

$$\mu := = \frac{1}{3} \min_{1 \le i, j, k \le n}$$

R

+

∩

{

d

(

−

→

s

i

,

p

j

)

,

d

(

−

→

s

i

,

−

→

s

j

∩

–

→

S

k

)

}

▪

Proof. In order to guarantee that the Minkowski sums are disjunct it suffices to show that μ is a lower bound of a third of the minimum of all pairwise infimum distances of disjunct traces si and sj.

证明。为了保证 Minkowski 和是不相交的，只需证明 μ 是不相交轨迹 si 和 sj 的所有成对下确界距离的最小值的三分之一的下界。

Constructing the straight skeleton is P-complete

3.4.3 Constructing the straight skeleton is P-complete

3.4.3 构建直线骨架是P-完全的

The concept of P-completeness is similar to the concept of NP-completeness. A problem A from P is said to be P-complete under NC-reductions if any problem in P can be reduced to A in NC time. The complexity class NC comprises all problems that can be efficiently solved in parallel. That is, they can be solved in poly-logarithmic time complexity using a polynomial number of processors. Hence, all P-complete problems cannot be efficiently solved using parallel computers, unless NC = P. In other words, P-complete problems are inherently sequential, provided that NC □= P. If the latter condition would be wrong then every problem solvable in polynomial time could also be solved efficiently in parallel. To show this it would suffice to find a single P-complete problem that can be solved in NC time. However, is is commonly assumed that NC □= P.

P-完备性的概念类似于NP-完备性的概念。如果P中的任何问题都可以在NC时间内归约到A，则称P中的问题A在NC归约下是P-完备的。复杂度类NC包含所有可以并行有效解决的问题。也就是说，它们可以在多对数时间复杂度内使用多项式数量的处理器来解决。因此，除非NC = P，否则所有P-完备问题都无法使用并行计算机有效解决。换句话说，如果NC □= P，则P-完备问题本质上是顺序的。如果后一个条件是错误的，那么每个可以在多项式时间内解决的问题也可以并行有效地解决。为了证明这一点，找到一个可以在NC时间内解决的P-完备问题就足够了。然而，通常假设NC □= P。

In order to show that a specific problem A is P-complete it suffices to reduce any Pcomplete problem to A in NC time. Alternatively, one could also seek for an according LOGSPACE reduction, because LOGSPACE ⊂ NC. For further details on P-completeness we refer to the book by Greenlaw, Hoover and Ruzzo [GHR95].

为了证明一个特定的问题A是P完全的，只需将任何P完全问题在NC时间内归约到A即可。或者，也可以寻找相应的LOGSPACE归约，因为LOGSPACE ⊂ NC。关于P完全性的更多细节，请参考Greenlaw、Hoover和Ruzzo的著作[GHR95]。

Atallah et al. [ACG93] described a framework for geometric reductions of the P-complete Planar Circuit Value problem and used it to prove the P-completeness of several geometric problems. For their framework, they consider specific binary circuits which are organized in layers: an input layer at the top and alternating routing layers and logic layers from top to bottom. The routing layers and logic layers consist of rows of components occupying nonoverlapping rectangles. A logic layer consists of a row of ∨-gates and a row of ¬-gates and a routing layer comprises rows of routing components, namely left shifts, right shifts, fan-out gates and vertical wires. The binary circuit has a

single output gate and the Circuit Value problem asks for the output value of this gate when a specific input vector is presented to the input gates. Atallah et al. proved that the Circuit Value problem for circuits of such a specific layout is still P-complete.

Atallah等人[ACG93]描述了一个用于平面电路值问题的几何归约框架，并用它来证明了几个几何问题的P完全性。对于他们的框架，他们考虑了以层组织的特定二元电路：顶部的输入层和从上到下交替的路由层和逻辑层。路由层和逻辑层由占据非重叠矩形的组件行组成。一个逻辑层由一排∨门和一排¬门组成，一个路由层由路由组件行组成，即左移、右移、扇出门和垂直线。二元电路有一个单一的输出门，电路值问题询问当一个特定的输入向量被呈现给输入门时，这个门的输出值。Atallah等人证明了对于具有这种特定布局的电路，电路值问题仍然是P-完全的。
Investigating the P-completeness of geometric problems often requires the availability of exact geometric computations, which are not in NC. For instance, they mention the problem of determining whether four points lie on a circle, which is an essential predicate when computing Voronoi diagrams. In order to investigate the P-completeness of geometric problems, Attalah et al. [ACG93] propose that the answers to basic geometric queries are provided by an oracle.

研究几何问题的P-完全性通常需要精确的几何计算，而这些计算并不属于NC。例如，他们提到了确定四个点是否共圆的问题，这在计算Voronoi图时是一个必要的谓词。为了研究几何问题的P-完全性，Attalah等人[ACG93] 提出基本几何查询的答案由预言机提供。

A basic building block for showing that the straight skeleton is P-complete is the construction of the triangles Δi. Assume pi, αi, vi, μ, and λ are given. We further assume that an oracle determines the intersection points of two circles with given centers and radii. Then, we can construct Δi as follows. We first compute the point $qi = pi + \lambda vi$, which is the position of mi at time one, see Figure 48. Then we construct the circle C1 with [pi, qi] as diameter and the circle C2 centered at pi with radius 1. The two circles C1, C2 intersect at two points, say ai, bi. The triangle $\Delta i*$ with vertices ai, bi, qi is an isosceles triangle with angle $2\alpha i$ at qi and therefore similar to Δi. (Note that pi, ai, qi form a right-angled triangle within Thales' circle C1.) The length of the arms of $\Delta i*$ at qi are at most $\lambda|vi|$. By scaling the triangle by the factor $\mu/2\lambda|vi|$ and by translating it accordingly, we get a triangle with the desired geometry. (Strictly speaking, the arms of the constructed triangle are a bit shorter than $\mu/2$, but this is only to our advantage.)

证明直线骨架是 P-完全问题的一个基本构建块是三角形 Δi 的构造。假设 pi, αi, vi, μ 和 λ 已知。我们进一步假设一个预言机确定了具有给定中心和半径的两个圆的交点。然后，我们可以如下构造 Δi。我们首先计算点 $qi = pi + \lambda vi$，这是 mi 在时间 1 的位置，参见图 48。然后我们构造以 [pi, qi] 为直径的圆 C1 和以 pi 为中心，半径为 1 的圆 C2。两个圆 C1, C2 相交于两点，比如 ai, bi。顶点为 ai, bi, qi 的三角形 $\Delta i*$ 是一个等腰三角形，在 qi 处的角度为 $2\alpha i$，因此与 Δi 相似。（注意，pi, ai, qi 在塔莱斯圆 C1.) 内形成一个直角三角形。）$\Delta i*$ 在 qi 处的臂的长度最多为 $\lambda|vi|$。通过将三角形按因子 $\mu/2\lambda|vi|$ 缩放并进行相应的平移，我们得到一个具有所需几何形状的三角形。（严格来说，构造的三角形的臂比 $\mu/2$ 稍短，但这只会对我们有利。）

Eppstein and Erickson [EE99] proved that the computation of the motorcycle graph is P-complete by presenting a LOGSPACE-reduction of the Circuit Value problem to the computation of the motorcycle graph. Eppstein and Erickson demonstrated how to translate the Circuit Value problem to the motorcycle graph construction problem by simulating each gadget of the binary circuit using motorcycles. The values 1 and 0 on a wire are represented by the presence or absence of a motorcycle on a track. The original question for the output value of a particular gate of the circuit can be translated to the question whether a specific motorcycle crashes until some distance from its start point. In other words, Eppstein and Erickson proved that the decision problem whether a specific motorcycle crashes until some distance from its start point is P-complete.

Eppstein和Erickson [EE99] 证明了摩托车图的计算是P-完全的，他们通过将电路值问题LOGSPACE归约到摩托车图的计算来完成证明。Eppstein和Erickson展示了如何通过使用摩托车模拟二进制电路的每个小工具，将电路值问题转化为摩托车图构造问题。导线上的值1和0分别由轨道上是否存在摩托车来表示。关于电路特定门输出值的原始问题可以转化为关于特定摩托车是否在距离其起点一定距离之前发生碰撞的问题。换句话说，Eppstein和Erickson证明了特定摩托车是否在距离其起点一定距离之前发生碰撞的决策问题是P-完全的。

Lemma 3.10. The construction of the straight skeleton of a planar straight-line graph is P-complete under LOGSPACE-reductions.

引理 3.10. 平面直线图的直线骨架的构造在 LOGSPACE 归约下是 P-完全的。

Proof. Eppstein and Erickson reduced the Circuit Value problem to a specific motorcycle graph problem. The next step is to reduce the motorcycle graph problem to the straightskeleton problem: we construct a suitable input graph G that allows us to apply Lemma 3.8 in order to decide whether a specific motorcycle crashes until some distance from its start point.

证明。Eppstein和Erickson将电路值问题简化为一个特定的摩托车图问题。下一步是将摩托车图问题简化为直线骨架问题：我们构造一个合适的输入图G，使我们能够应用引理3.8，以确定一辆特定的摩托车是否会碰撞，直到距离其起点一定距离。

According to [EE99] all O(1) different types of motorcycle gadgets are arranged in an n × n grid. Each gadget takes constant space and consists of O(1) motorcycles. To determine a sufficiently large λ, we need bounds on L, Φ and μ. An upper bound on L is the length of the diagonal of the n × n grid. Further, sin Φ ≥ 1/2 since the direction angles of the motorcycles are all multiples of π/4. A lower bound on μ can be obtained by applying Lemma 3.9 on each gadget independently and taking the minimum among them. Finally, we build G by modeling each motorcycle (independently from each other) as an isosceles triangle, as described in Section 3.4.1.

根据[EE99]，所有O(1)种不同类型的摩托车装置都排列在一个n × n的网格中。每个装置占用恒定空间，并由O(1)辆摩托车组成。为了确定一个足够大的λ，我们需要L、Φ和μ的界限。 L的上界是n × n网格的对角线的长度。此外，由于摩托车的方向角都是π/4的倍数，因

此sin Φ ≥ 1/2。通过对每个装置独立应用引理3.9，并在它们之间取最小值，可以获得μ的下界。最后，我们通过将每辆摩托车（彼此独立）建模为一个等腰三角形来构建G，如第3.4.1节所述。
We can easily extend the construction of G to form a polygon with holes, by adding a sufficiently large bounding box to G. As remarked in [EE99], only one motorcycle m may leave the bounding box B of the n × n grid. The motorcycle m encodes the output of the binary circuit by leaving B if the circuit evaluates to 1 and by crashing within B if the circuit evaluates to 0.

通过向G添加一个足够大的边界框，我们可以很容易地扩展G的构造，以形成一个带孔的多边形。正如[EE99]中所述，只有一个摩托车m可以离开n × n网格的边界框B。摩托车m通过离开B来编码二进制电路的输出（如果电路的计算结果为1），并通过在B内崩溃来编码二进制电路的输出（如果电路的计算结果为0）。

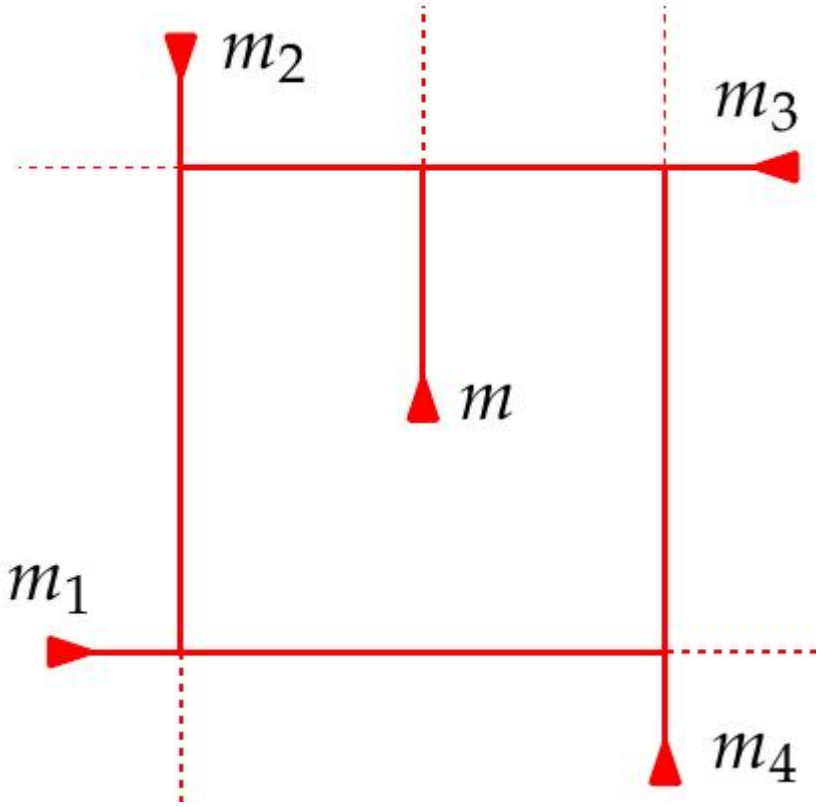


Figure 53: These motorcycle traces cannot be approximated easily by a straight skeleton of a simple polygon.

图 53：这些摩托车轨迹不容易用简单多边形的直线骨架来近似。

By Lemma 3.8 the reflex wavefront vertex v, which corresponds to m, encodes the output of the binary circuit by leading to a split event until time μ/4 if and only if the circuit evaluates to 0. Lemma 3.4 implies that the wavefront vertices stay within B + Dμ, except possibly v. Hence, we could enlarge B by 2μ at each side and add it to G such that the wavefronts of B do not interfere with the wavefronts of the triangles until time μ/4, except for v. Still, we can determine the output of the binary circuit by checking whether the reflex

straight-skeleton arc that corresponds to v, ends within B + Dμ until time μ/4. (Recall that the end of a reflex straight-skeleton arc marks the place where the reflex wavefront vertex led to a split event.)

根据引理3.8，对应于m的反射波前顶点v，通过导致一个分裂事件来编码二元电路的输出，当且仅当电路的计算结果为0时，该分裂事件发生在时间μ/4之前。引理3.4表明，波前顶点保持在B + Dμ内，v可能除外。因此，我们可以将B的每一侧扩大2μ，并将其添加到G中，使得B的波前在时间μ/4之前不会干扰三角形的波前，v除外。不过，我们仍然可以通过检查对应于v的反射直线骨架弧是否在时间μ/4之前结束于B + Dμ内，来确定二元电路的输出。（回想一下，反射直线骨架弧的末端标志着反射波前顶点导致分裂事件发生的位置。）
Corollary 3.11. The construction of the straight skeleton of a polygon with holes is P-complete under LOGSPACE-reductions.

推论 3.11. 在 LOGSPACE 归约下，带孔多边形的直骨架构造是 P-完全的。

Unfortunately, our P-completeness proof cannot be applied easily to simple polygons. Consider the five motorcycles depicted in Figure 53. A simple polygon, whose straight skeleton would approximate the motorcycle traces, would need to connect the start points of m and m1, . . . , m4. But in order to decide where the red square, formed by the traces of m1, . . . , m4, can be penetrated by the polygon, while avoiding to stop a motorcycle too early, it would be necessary to know that a specific motorcycle mi crashes into a specific motorcycle mj. However, deciding whether a specific motorcycle crashes does not seem much easier than computing the whole motorcycle graph. Hence, it remains open whether the computation of straight skeletons of polygons is P-complete.

遗憾的是，我们的 P-完备性证明不能轻易地应用于简单多边形。考虑图 53 中描绘的五辆摩托车。一个简单的多边形，其直线骨架将近似于摩托车的轨迹，需要连接 m 和 m1, . . . , m4 的起始点。但是，为了确定由 m1, . . . , m4 的轨迹形成的红色正方形可以被多边形穿透的位置，同时避免过早地停止一辆摩托车，有必要知道特定的摩托车 mi 是否撞向特定的摩托车 mj。然而，确定一辆特定的摩托车是否会相撞似乎并不比计算整个摩托车图更容易。因此，多边形直线骨架的计算是否是 P-完备的仍然是一个未解决的问题。
The investigations done in this thesis are driven by the lack of an efficient implementation of straight skeletons for real-world purposes that stands opposite to the large number of different industrial and academical applications. At the moment, the state-of-the-art straightskeleton implementation is shipped with the CGAL library. However, our experiments illustrate that the CGAL implementation is only applicable to real-world problems in a limited manner: Both, the runtime and the memory footprint of the CGAL implementation increase at least quadratically with the input size. While time is more or less an infinite resource, the opposite is true for space. The quadratic memory consumption is an issue that basically renders the implementation in CGAL inapplicable to datasets containing more than ten thousand vertices.

本论文中的研究是由缺乏适用于实际应用的直线骨架的高效实现所驱动的，这与大量不同的工业和学术应用形成了对比。目前，最先进的直线骨架实现随 CGAL 库一起发布。然而，我们的实验表明，CGAL 的实现在实际问题中的适用性有限：CGAL 实现的运行时间

和内存占用至少以输入大小的二次方增长。虽然时间或多或少是一种无限的资源，但空间的情况恰恰相反。二次内存消耗是一个问题，它基本上使得 CGAL 中的实现在包含超过一万个顶点的数据集上无法应用。
We started our investigations towards an efficient straight-skeleton algorithm with an analysis of the triangulation-based approach by Aichholzer and Aurenhammer [AA98] in Section 2.2. The gap between Ω(n2) and O(n3) for the worst-case number of flip events leads to an open question regarding the time complexity of the algorithm. We present different results regarding this gap and finally prove the existence of Steiner triangulations that are free of flip-events. This result is primarily of theoretical interest, since we use the straight skeleton for the placement of the Steiner points. However, the fact that flip-event-free Steiner triangulations exist led to a novel wavefront-type straight-skeleton algorithm for simple nondegenerate polygons which is based on the motorcycle graph in Section 2.3.

我们从分析 Aichholzer 和 Aurenhammer 在第 2.2 节中提出的基于三角剖分的方法 [AA98] 开始了对高效直骨骼算法的研究。最坏情况下翻转事件数量的 Ω(n2) 和 O(n3) 之间的差距引出了关于该算法时间复杂度的未决问题。我们展示了关于这个差距的不同结果，并最终证明了无翻转事件的斯坦纳三角剖分的存在。这个结果主要具有理论意义，因为我们使用直骨骼来放置斯坦纳点。然而，无翻转事件的斯坦纳三角剖分的存在导致了一种新的波前型直骨骼算法，该算法适用于简单的非退化多边形，该算法基于第 2.3 节中的摩托车图。

In order to make this algorithm applicable to real-world input we needed to remove the non-degeneracy assumption on the input. In Section 2.4, we carefully generalized the motorcycle graph in order to reflect so-called vertex events for straight skeletons. That is, the simultaneous crash of two or more motorcycles into each other may require to start an additional motorcycle. We demand two essential geometric properties for the generalized motorcycle graph: (i) M(G) needs to cover the reflex arcs of S (G), also in the presence of vertex events, and (ii) G + M(G) needs to induce a convex tessellation of the plane. These two properties allowed us to extend our algorithm to arbitrary planar straight-line graphs G in Section 2.5. Furthermore, the generalization of the motorcycle graph leads to an alternative characterization of the straight skeleton of arbitrary planar straight-line graphs by extending the characterization of Cheng and Vigneron and it motivated a straight-skeleton algorithm that is based on 3D graphics hardware.

为了使该算法适用于真实世界的输入，我们需要移除对输入的非退化假设。在第2.4节中，我们仔细地推广了摩托车图，以反映直线骨架的所谓顶点事件。也就是说，两辆或多辆摩托车同时相互碰撞可能需要启动额外的摩托车。我们对广义摩托车图提出了两个重要的几何性质要求：（i）即使在存在顶点事件的情况下，M(G)也需要覆盖S(G)的反射弧，并且（ii）G + M(G)需要诱导平面的凸面镶嵌。这两个性质使我们能够在第2.5节中将我们的算法扩展到任意平面直线图G。此外，摩托车图的推广通过扩展Cheng和Vigneron的表征，导致了对任意平面直线图的直线骨架的另一种表征，并且它激发了一种基于3D图形硬件的直线骨架算法。
The wavefront-type straight-skeleton algorithm presented in Section 2.5 is easy to implement, has a worst-case runtime of O(n2 log n) and operates in O(n) space. The resulting implementation Bone uses ordinary double-precision floating-point arithmetic

and accepts planar straight-line graphs as input. Our experiments showed an O(n log n) runtime on 13 500 datasets of different types. This constitutes an improvement of a linear factor in time and space compared to the implementation in CGAL, which only accepts polygons with holes as input. On datasets with several thousand vertices, our implementation is up to two orders of magnitude faster and we need less than 100 megabytes of memory instead of several gigabytes. Moreover, due to our low memory requirements, we are able to compute the straight skeleton of datasets with a million vertices. These circumstances make Bone the fastest straight-skeleton implementation at the moment and the first extensively tested implementation that is capable to process planar straight-line graphs originating from realworld applications.

第2.5节中介绍的波前型直线骨架算法易于实现，最坏情况下的运行时间为O(n2 log n)，并在O(n)空间中运行。由此产生的实现Bone使用普通的双精度浮点运算，并接受平面直线图作为输入。我们的实验表明，在13 500个不同类型的数据集上，运行时间为O(n log n)。与CGAL中的实现相比，这在时间和空间上都改进了一个线性因子，CGAL仅接受带孔多边形作为输入。在具有数千个顶点的数据集上，我们的实现速度提高了两个数量级，并且我们需要的内存少于100兆字节，而不是几千兆字节。此外，由于我们对内存的要求较低，因此我们能够计算具有一百万个顶点的数据集的直线骨架。这些情况使得Bone成为目前最快的直线骨架实现，也是第一个经过广泛测试的、能够处理来自真实世界应用程序的平面直线图的实现。
In order to push Bone to industrial strength, we plan to enhance the numerical stability in the presence of parallel wavefronts that collapse simultaneously. Detecting the simultaneous collapse of larger parts of the wavefront is simplified by exploiting the convex tessellation induced by the motorcycle graph. From an abstract point of view we gain from the fact that the motorcycle graph already provides a sufficient amount of information on the topology of the straight skeleton. Bone is currently extended to use different numerical backends by colleagues, including the arbitrary precision library MPFR [MPF] and the exact geometric computation library CORE [Cor]. The necessary work for MPFR is almost finished and preliminary runtime tests with a floating-point precisions of 212 and 1000 bits show a drop in performance by a factor of approximately 10–20, which is still reasonable for real-world applications.

为了将Bone推向工业强度，我们计划增强在并行波前同时坍塌时数值的稳定性。利用摩托车图所诱导的凸镶嵌简化了对波前较大区域同时坍塌的检测。从抽象的角度来看，我们受益于摩托车图已经提供了关于直线骨架拓扑的足够信息。Bone目前正在由同事扩展，以使用不同的数值后端，包括任意精度库MPFR [MPF]和精确几何计算库CORE [Cor]。MPFR的必要工作几乎已经完成，使用212和1000位浮点精度的初步运行时测试表明，性能下降了大约10-20倍，对于实际应用来说，这仍然是合理的。

Even though Bone exhibits an O(n log n) runtime in practice, the gap between the lower bound of Ω(n log n) and the theoretically fastest algorithm by Eppstein and Erickson [EE99], with a theoretical worst-case time complexity of O(n17/11+ϵ), remains open. If the motorcycle graph M(G) is known, the algorithm behind Bone has an O((n + k) log n) time complexity, where k ∈ O(nr) denotes the number of switch events and r ∈ O(n) denotes the number of reflex wavefront vertices. In order to obtain a worst-case runtime

of O(n log n) it would be necessary to bound the number k of switch events to O(n). One approach towards this goal could be the introduction of additional motorcycles that are launched from convex wavefront vertices of W (G, 0) such that the number of interactions between moving Steiner vertices and convex wavefront vertices is reduced to O(n). However, in order to extend the basic algorithm behind Bone accordingly, we would require that any extension of the motorcycle graph M(G) by additional motorcycles needs to maintain the validity of Lemma 2.25 and Theorem 2.26: G + M(G) needs to induce a convex tessellation and the motorcycle traces need to cover the reflex straight-skeleton arcs.

尽管 Bone 在实践中表现出 O(n log n) 的运行时间，但 Ω(n log n) 的下界与 Eppstein 和 Erickson [EE99] 的理论上最快算法之间的差距仍然存在，该算法的理论最坏情况时间复杂度为 O(n17/11+ϵ)。如果摩托车图 M(G) 是已知的，则 Bone 背后的算法具有 O((n + k) log n) 的时间复杂度，其中 k ∈ O(nr) 表示切换事件的数量，r ∈ O(n) 表示反射波前顶点的数量。为了获得 O(n log n) 的最坏情况运行时间，有必要将切换事件的数量 k 限制为 O(n)。实现此目标的一种方法可能是引入额外的摩托车，这些摩托车从 W (G, 0) 的凸波前顶点发射，从而将移动 Steiner 顶点和凸波前顶点之间的交互次数减少到 O(n)。然而，为了相应地扩展 Bone 背后的基本算法，我们需要通过额外的摩托车对摩托车图 M(G) 的任何扩展都需要保持引理 2.25 和定理 2.26 的有效性：G + M(G) 需要导出凸镶嵌，并且摩托车轨迹需要覆盖反射直线骨架弧。

We also look forward to generalize the algorithms behind Bone in order to compute weighted straight skeletons. In principal, wavefront-type algorithms tend to have a straightforward generalization from the unweighted straight skeleton to the weighted counterpart. However, in order to generalize Bone to weighted straight skeletons, it is necessary to adapt the definition of the motorcycle graph such that Lemma 2.25 and Theorem 2.26 still hold in the weighted case. First of all, the speed and direction of each motorcycle is given by Lemma 1.11 in order to match the speeds of the corresponding reflex wavefront vertices. Assume for a moment that we simply transfer the rules for launching new motorcycles from Section 2.4.1. If we do so, Lemma 2.25 remains true for the Case (d) in Figure 31, but not for Case (c). Note that if a vertex event happens then the reflex straight-skeleton arcs do not need to tessellate a local disk into convex slices. That is, the lower chains of the faces do not need to be convex. In fact, the faces do not even need to be monotone. In order to fix Lemma 2.25, we could launch a second motorcycle that continues the movement of the right ancestor, as in Case (d). However, the question remains whether the number of motorcycles is still in O(n) after this adaption. In the next step, we need to guarantee that Theorem 2.26 remains true for this generalized motorcycle graph. Note that again the tilted motorcycle traces mˆ 1, . . . , mˆ k with the same right arm e lie on the supporting plane of the tilted face fˆ(e) and at least the Step (i) in the proof of Theorem 2.26 remains true. We also want to note that a generalization of Theorem 2.26 could also provide a lower envelope characterization of the weighted straight skeleton, which does not exist so far. Recall that Eppstein and Erickson [EE99] showed that the trivial generalization does not work, see Figure 16.

我们还期望推广 Bone 背后的算法，以便计算加权直线骨架。原则上，波前型算法倾向于从非加权直线骨架到加权直线骨架进行直接推广。然而，为了将 Bone 推广到加权直线骨架，有必要调整摩托车图的定义，使得引理 2.25 和定理 2.26 在加权情况下仍然成立。首先，每辆摩托车的速度和方向由引理 1.11 给出，以便与相应凹波前顶点的速度相匹配。假设我们暂时简单地转移第 2.4.1 节中启动新摩托车的规则。如果我们这样做，引理 2.25 对于图 31 中的情况 (d) 仍然成立，但对于情况 (c) 不成立。请注意，如果发生顶点事件，则凹直线骨架弧不需要将局部圆盘细分为凸切片。也就是说，面的下链不需要是凸的。事实上，这些面甚至不需要是单调的。为了修正引理 2.25，我们可以启动第二辆摩托车，继续右祖先的移动，如情况 (d) 所示。然而，问题仍然是，经过这种调整后，摩托车的数量是否仍然在 O(n) 范围内。在下一步中，我们需要保证定理 2.26 对于这种广义摩托车图仍然成立。请注意，倾斜的摩托车轨迹 mˆ 1, . . . , mˆ k 再次具有相同的右臂 e，位于倾斜面 fˆ(e) 的支撑平面上，并且至少定理 2.26 证明中的步骤 (i) 仍然成立。我们还要指出，定理 2.26 的推广也可以提供加权直线骨架的下包络表征，而这目前尚不存在。回想一下，Eppstein 和 Erickson [EE99] 表明，简单的推广是行不通的，参见图 16。
In Chapter 3 we took a closer look at the motorcycle graph. In order to compute straight skeletons fast in practice by means of Bone, we require an implementation for the generalized motorcycle graph that performs well on real-world data. We started with a stochastic analysis of the average trace length in a motorcycle graph in Section 3.2. It turned out that if the motorcycles are distributed uniformly within the unit square then we can expect an average trace length of Θ(1/√n). In other words, if we impose a regular √n × √n grid on the unit square then we expect that a motorcycle crosses O(1) grid cells on average. This fact motivated a simple pragmatic approach: we enhanced the straight-forward approach based on a priority-queue with geometric hashing in Section 3.3. This measurement reduces the runtime from O(n2 log n) to O(n log n) for input data where the motorcycles are distributed uniformly enough. We performed extensive runtime tests with our implementation Moca. For the vast majority of our datasets Moca runs in O(n log n) time, which is one of the key reasons for the good overall performance of our straight-skeleton code Bone. Finally, we also used Moca in order to substantiate the theoretical predictions provided by the stochastic analysis.

在第3章中，我们更仔细地研究了摩托车图。为了通过Bone在实践中快速计算直线骨架，我们需要一个广义摩托车图的实现，该实现在真实世界的数据上表现良好。我们从第3.2节中摩托车图中平均轨迹长度的随机分析开始。结果表明，如果摩托车在单位正方形内均匀分布，那么我们可以预期平均轨迹长度为Θ(1/√n)。换句话说，如果我们在单位正方形上施加一个规则的√n × √n网格，那么我们预计一辆摩托车平均穿过O(1)个网格单元。这一事实促使我们采用了一种简单的实用方法：我们在第3.3节中基于优先级队列的方法，通过几何哈希进行了增强。对于摩托车分布足够均匀的输入数据，这种测量将运行时间从O(n2 log n)减少到O(n log n)。我们使用我们的实现Moca进行了广泛的运行时测试。对于我们绝大多数数据集，Moca在O(n log n)时间内运行，这是我们的直线骨架代码Bone良好整体性能的关键原因之一。最后，我们还使用Moca来证实随机分析提供的理论预测。

We plan to further boost the performance of Moca by implementing the O(n log n) algorithm to compute the motorcycle graph outside the convex hull of the start points, see Section 3.3.4. Furthermore, in order to make the runtime performance of Moca more robust against clustered start points, one may also replace the ordinary rectangular grid

by quad trees, which allows a non-uniform tessellation of the plane. Additionally, quad trees would enable us to dynamically increase the tessellation depth at regions where motorcycles accumulate during the propagation process.

我们计划通过实施 O(n log n) 算法来计算起始点凸包外部的摩托车图，从而进一步提升 Moca 的性能，参见第 3.3.4 节。此外，为了使 Moca 的运行时性能对聚集的起始点更加稳健，还可以用四叉树代替普通的矩形网格，这允许平面的非均匀镶嵌。此外，四叉树将使我们能够动态地增加摩托车在传播过程中积累的区域的镶嵌深度。
Our final contribution concerns the geometric relation of motorcycle graphs and straight skeletons. By Theorem 2.26 we know that the reflex straight-skeleton arcs of S (G) approximate the motorcycle traces of M(G) up to a specific extent. Furthermore, one observers that in general the gap between the reflex arcs and the motorcycle traces decreases if the speed of the motorcycle increases. In Section 3.4, we first show that we can construct a planar straight-line graph G whose straight skeleton approximates the motorcycle graph. Based on this result, we present a simple algorithm that computes the motorcycle graph using the straight skeleton. Finally, we show that the resulting algorithm admits a LOGSPACEreduction of the motorcycle graph problem to the straight-skeleton problem. Consequently, the P-completeness of the motorcycle graph by Eppstein and Erickson [EE99] implies the P-completeness of the straight-skeleton problem for planar straight-line graphs and for polygons with holes. We want to note that Eppstein and Erickson [EE99] were the first to mention that straight skeleton is P-complete, but no proof was given. The P-completeness of straight skeletons has important practical implications: no efficient parallel algorithms exist to compute straight skeletons of planar straight-line graphs and polygons with holes, provided that P □= NC. We also note that it is desirable to find an efficient sequential reduction of motorcycle graphs to straight skeletons in order to transfer lower bounds from motorcycle graphs to straight skeletons in the sequential manner.

我们最后的贡献涉及摩托车图和直线骨架的几何关系。根据定理2.26，我们知道S (G)的反射直线骨架弧在一定程度上近似于M(G)的摩托车轨迹。此外，观察到通常情况下，如果摩托车的速度增加，反射弧和摩托车轨迹之间的差距会减小。在第3.4节中，我们首先证明我们可以构造一个平面直线图G，其直线骨架近似于摩托车图。基于这个结果，我们提出了一个简单的算法，该算法使用直线骨架计算摩托车图。最后，我们证明了该算法允许将摩托车图问题LOGSPACE归约到直线骨架问题。因此，Eppstein和Erickson [EE99]提出的摩托车图的P-完全性意味着平面直线图和带孔多边形的直线骨架问题的P-完全性。我们想指出的是，Eppstein和Erickson [EE99]是第一个提到直线骨架是P-完全的，但没有给出证明。直线骨架的P-完全性具有重要的实际意义：不存在有效的并行算法来计算平面直线图和带孔多边形的直线骨架，前提是P □= NC。我们还注意到，为了以顺序方式将摩托车图的下界转移到直线骨架，需要找到一个将摩托车图有效顺序归约到直线骨架的方法。

pq, e, fˆ(e) The supporting line of the points p, q resp. the edge e and the supporting plane of the lifted face fˆ(e).

A + B The Minkowski sum A + B = {x + y : x ∈ A, y ∈ B} for two point sets A and B.

A + B 闵可夫斯基和 A + B = {x + y : x ∈ A, y ∈ B}，对于两个点集 A 和 B。

[pq] The straight-line segment between two points p and q.

aˆ, sˆ, mˆ , fˆ(e) The lifted counterpart of the arc a, motorcycle trace s, motorcycle m, straight-skeleton face f (e) in the terrain model.

d(p, q) The Euclidean distance between two points p and q.

d(p, q) 点p和q之间的欧几里得距离。

d(A, B) The infimum distance infx∈A,y∈B d(x, y) for two point sets A and B. Dr A disk with radius r and the origin as center.

d(A, B) 两个点集A和B的下确界距离infx∈A,y∈B d(x, y)。Dr 一个以原点为中心，半径为r的圆盘。

e(t) The union of the straight-line segments occupied by the wavefront edge e at time t, see Definition 2.2.

e(t) 时刻 t 时波前边缘 e 所占据的直线段的并集，参见定义 2.2。

e(t) The supporting line of e(t). If e is emanated by a terminal vertex or an isolated vertex v of G then we define e(0) := limt↘0 e(t), see Definition 2.2.

e(t)：e(t) 的支撑线。如果 e 是由终端顶点或 G 的孤立顶点 v 发出的，那么我们定义 e(0) := limt↘0 e(t)，参见定义 2.2。

f (e) The straight-skeleton face of the wavefront edge e, see Definition 1.1. G A planar straight-line graph (with no isolated vertices).

f (e) 波前边缘 e 的直线骨架面，参见定义 1.1。G 平面直线图（没有孤立顶点）。

M(m1, . . . , mn) The motorcycle graph of the motorcycles m1, . . . , mn, see Definition 1.8.

M(m1, . . . , mn) 摩托车m1, . . . , mn的摩托车图，见定义1.8。

M(P) The motorcycle graph induced by the simple polygon P, see Definition 1.9.

M(P) 由简单多边形P导出的摩托车图，参见定义1.9。

M(G) The motorcycle graph induced by a planar straight-line graph G, see Definition 2.23.

M(G) 由平面直线图 G 导出的摩托车图，参见定义 2.23。

P A simple polygon in the plane (with holes if mentioned explicitly).

平面上的一个简单多边形（如果明确提及，则带有孔）。P

S (P) The straight skeleton of a simple polygon P, only considered within the polygon P, see Definition 1.1.

S(P) 简单多边形P的直骨架，仅在多边形P内考虑，参见定义1.1。
S (G) The straight skeleton of a planar straight-line graph G, see Section 1.2.2.

S(G) 平面直线图G的直骨架，参见第1.2.2节。

T (G) The terrain model corresponding to S (G), see Definition 1.5.

T (G) 对应于 S (G) 的地形模型，参见定义 1.5。

W (G, t) The wavefront of G at time t ≥ 0, see Definition 1.4.

W (G, t) 在时间 t ≥ 0 时 G 的波前，参见定义 1.4。

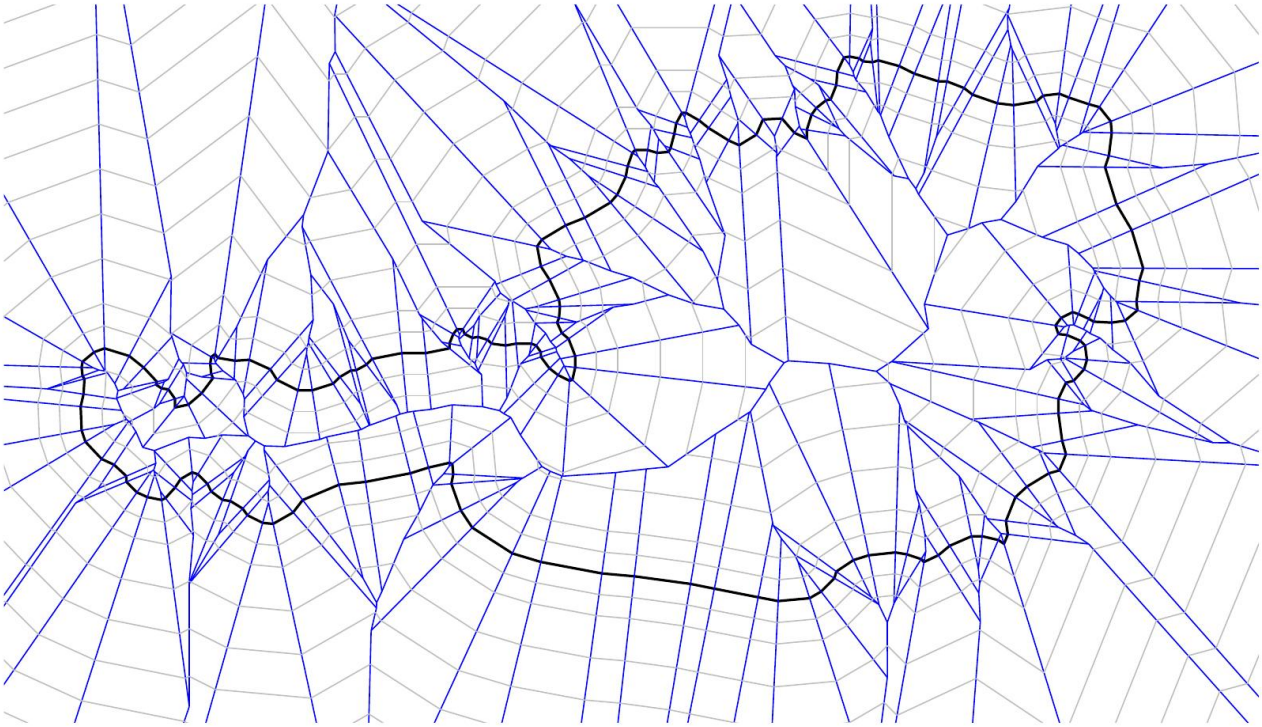


Figure 56: The straight skeleton and offset curves of the outline of Austria.

图 56：奥地利轮廓的直线骨架和偏移曲线。

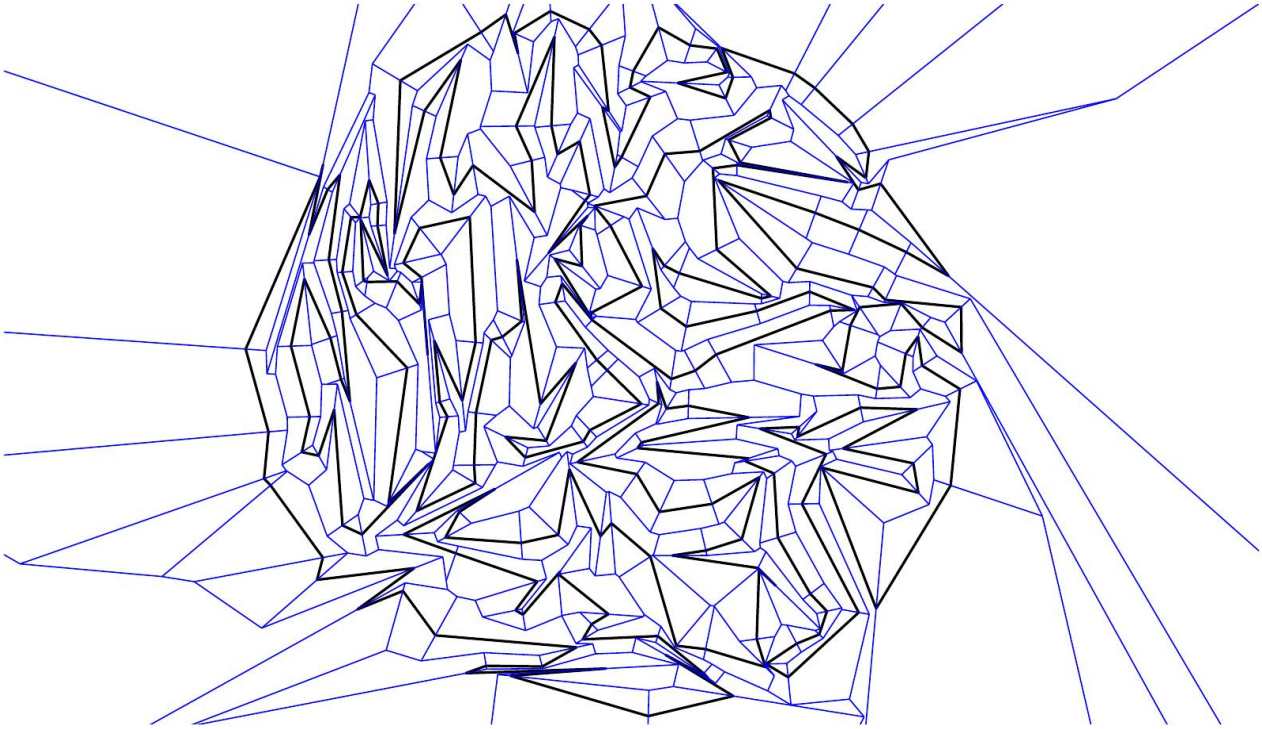


Figure 57: The straight skeleton of a random polygon generated by RPG.

图 57：由 RPG 生成的随机多边形的直线骨架。

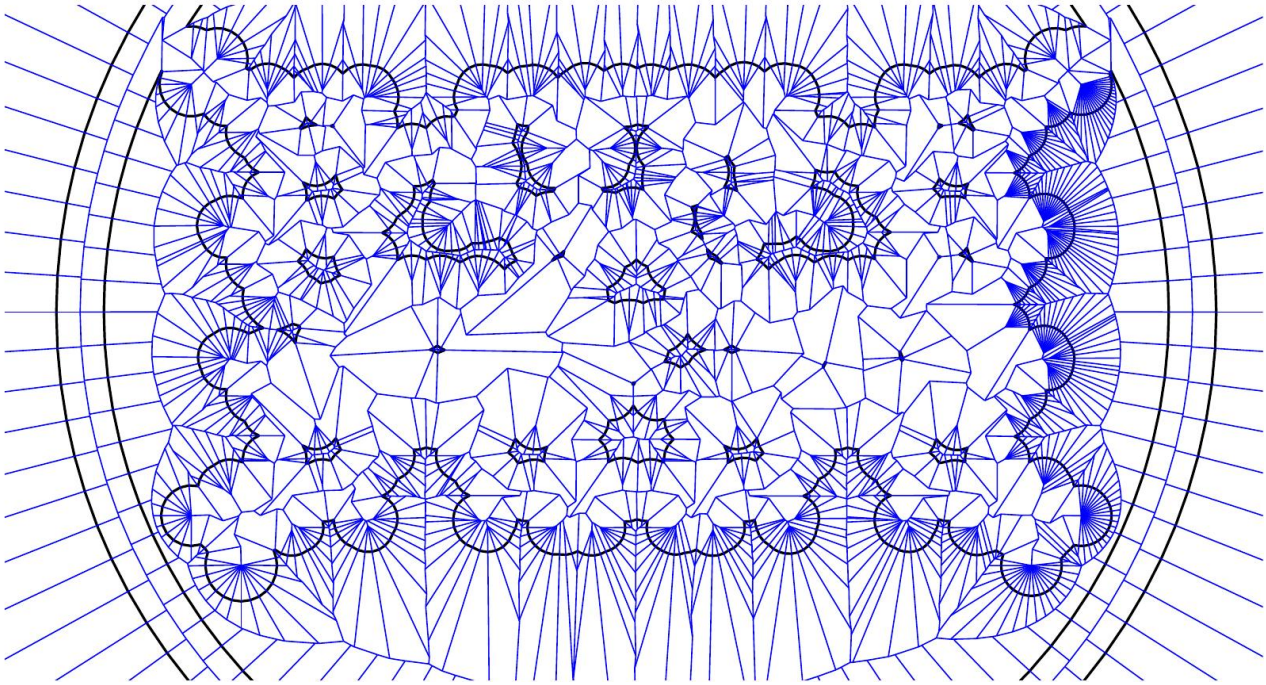


Figure 58: The straight skeleton of a polygon with holes.

图 58：带孔多边形的直线骨架。

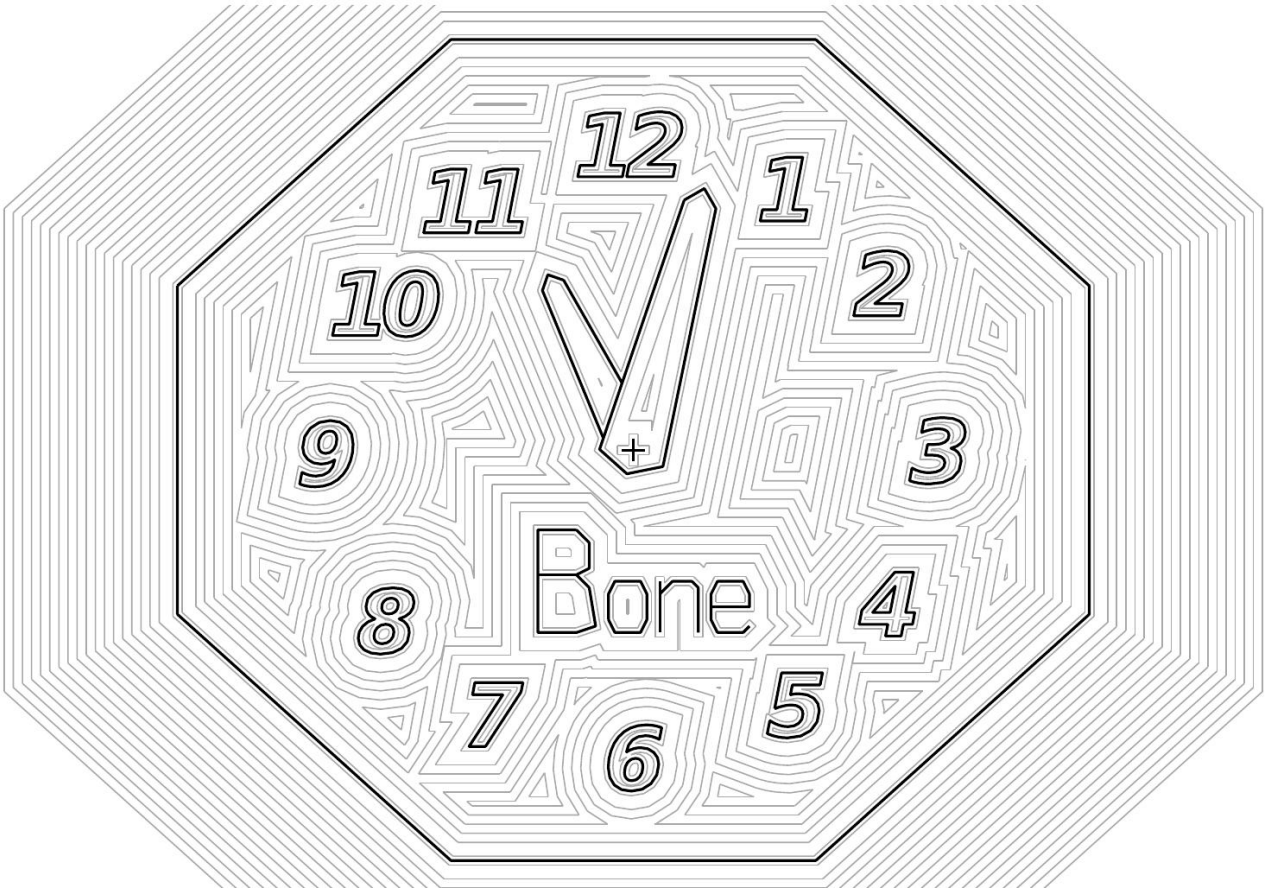


Figure 59: Offset curves based on the straight skeleton of a planar straight-line graph.

图 59：基于平面直线图的直线骨架的偏移曲线。

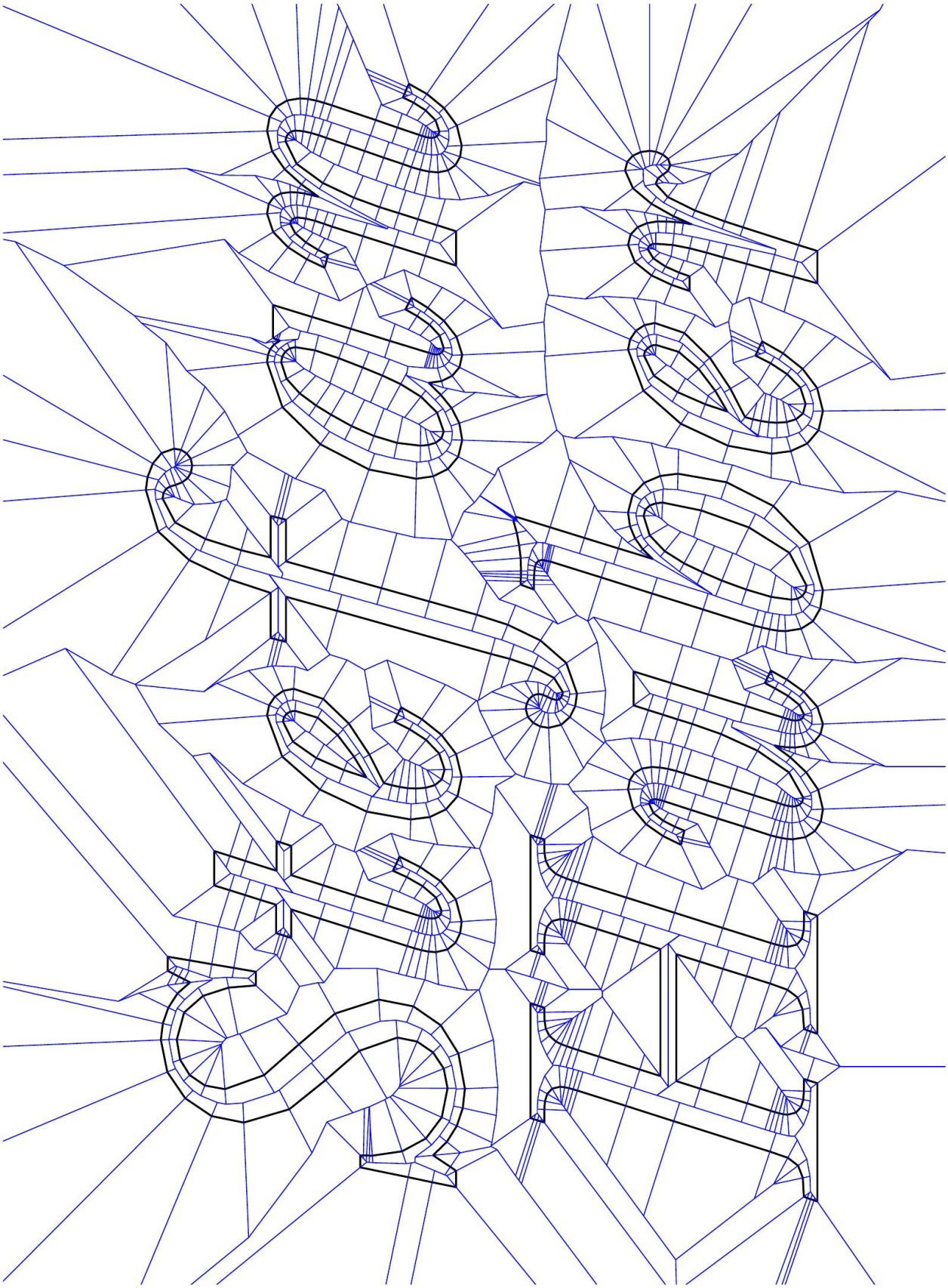


Figure 60: The straight skeleton of a font outline.

图 60：字体轮廓的直线骨架。

Salam
Hello

Figure 61: Offset curves of a font outline.

图 61：字体轮廓的偏移曲线。

B I B L I O G R A P H Y

参考文献

[AA96] O. Aichholzer and F. Aurenhammer. Straight Skeletons for General Polygonal Figures. In Proc. 2nd Annu. Internat. Conf. Comput. Combinatorics, volume 1090 of Lecture Notes Comput. Sci., pages 117–126. Springer-Verlag, 1996.

[AA98] O. Aichholzer and F. Aurenhammer. Straight Skeletons for General Polygonal Figures in the Plane. In A.M. Samoilenko, editor, Voronoi's Impact on Modern Science, Book 2, pages 7–21. Institute of Mathematics of the National Academy of Sciences of Ukraine, Kiev, Ukraine, 1998.

[AAAG95] O. Aichholzer, D. Alberts, F. Aurenhammer, and B. Gärtner. Straight Skeletons of Simple Polygons. In Proc. 4th Internat. Symp. of LIESMARS, pages 114–124, Wuhan, P.R. China, 1995.

[AAP04] O. Aichholzer, F. Aurenhammer, and B. Palop. Quickest Paths, Straight Skeletons, and the City Voronoi Diagram. Discrete Comput. Geom., 31(1):17–35, 2004.

[ACG93] M.J. Atallah, P.B. Callahan, and M.T. Goodrich. P-complete Geometric Problems. Internat. J. Comput. Geom. Appl., 3(4):443–462, 1993.

[AE99] P. K. Agarwal and J. Erickson. Geometric Range Searching and Its Relatives. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, Advances in Discrete and Computational Geometry, volume 223 of Contemporary Mathematics, pages 1–56. American Mathematical Society, 1999.

[AGSS87] A. Aggarwal, L.J. Guibas, J. Saxe, and P. Shor. A Linear Time Algorithm for Computing the Voronoi Diagram of a Convex Polygon. In Proc. 19th Annu. ACM Sympos. Theory Comput., pages 39–45, 1987.

[AH96] T. Auer and M. Held. Heuristics for the Generation of Random Polygons. In Proc. Canad. Conf. Comput. Geom. (CCCG'96), pages 38–44, Ottawa, Canada, Aug 1996. Carleton University Press.

[AM94] P. Agarwal and J. Matousek. ˇ On Range Searching with Semialgebraic Sets. Discrete Comput. Geom., 11:393–418, 1994.

[AS95] H. Alt and O. Schwarzkopf. The Voronoi Diagram of Curved Objects. In Proc. 11th Annu. ACM Sympos. Comput. Geom., pages 89–97, Vancouver, BC, Canada, 1995.

[BEGV08] G. Barequet, D. Eppstein, M. T. Goodrich, and A. Vaxman. Straight Skeletons of Three-Dimensional Polyhedra. In Proc. 16th Annu. Europ. Symp. Algorithms (ESA '08), pages 148–160, Karlsruhe, Germany, Sep 2008.

[BGLSS04] G. Barequet, M.T. Goodrich, A. Levi-Steiner, and D. Steiner. Contour Interpolation by Straight Skeletons. Graph. Models, 66(4):245–260, 2004.

[Ble] Blender. http://www.blender.org/. last checked on May, 2011.
[BO79] J. Bentley and T. Ottmann. Algorithms for Reporting and Counting Geometric Intersections. IEEE Trans. Comput., C-28:643–647, 1979.
[Cac04] F. Cacciola. A CGAL Implementation of the Straight Skeleton of a Simple 2D Polygon with Holes. In 2nd CGAL User Workshop, Polytechnic Univ., Brooklyn, New York, USA, June 2004.

[CEG+91] B. Chazelle, H. Edelsbrunner, M. Grigni, L. Guibas, J. Hershberger, M. Sharir, and J. Snoeyink. Ray Shooting in Polygons Using Geodesic Triangulations. In Proc. 18th Internat. Colloq. Automata Lang. Program., number 510 in Lecture Notes Comput. Sci., pages 661–673. Springer-Verlag, 1991.

[CGA] CGAL — Computational Geometry Algorithms Library. http://www.cgal. org. accessed on May, 2011.
[Cha91] B. Chazelle. Triangulating a Simple Polygon in Linear Time. Discrete Comput. Geom., 6:485–524, 1991.

[Cha93] B. Chazelle. Cutting Hyperplanes for Divide-and-Conquer. Discrete Comput. Geom., 9:145–158, Apr 1993.

[Cha04] B. Chazelle. Cuttings. In Handbook of Data Structures and Applications, pages 25.1–25.10. 1 edition, 2004. Chapman and Hall/CRC Press.

[Cor] CORE library project. http://cs.nyu.edu/exact/core_pages/. accessed on May, 2011.
[CRU89] J. Czyzowicz, I. Rival, and J. Urrutia. Galleries, Light Matchings and Visibility Graphs. In Proc. 1st Workshop Algorithms Data Struct., pages 316–324, Ottawa, Canada, Aug 1989.

[CSW99] F. Chin, J. Snoeyink, and C.A. Wang. Finding the Medial Axis of a Simple Polygon in Linear Time. Discrete Comput. Geom., 21(3):405–420, 1999.

[CV02] S.-W. Cheng and A. Vigneron. Motorcycle Graphs and Straight Skeletons. In Proc. 13th ACM-SIAM Sympos. Discrete Algorithms, pages 156–165, San Francisco, CA, USA, 2002.

[CV07] S.-W. Cheng and A. Vigneron. Motorcycle Graphs and Straight Skeletons. Algorithmica, 47:159–182, Feb 2007.

[DDL98] E. D. Demaine, M. L. Demaine, and A. Lubiw. Folding and Cutting Paper. In Revised Papers from the Japan Conference on Discrete and Computational Geometry (JCDCG'98), volume 1763 of Lecture Notes Comput. Sci., pages 104–117, Tokyo, Japan, Dec 1998.

[DDLS05] E. D. Demaine, M. L. Demaine, J. F. Lindy, and D. L. Souvaine. Hinged Dissection of Polypolyhedra. In Proc. 9th Workshop Algorithms Data Struct. (WADS 2005), volume 3608 of Lecture Notes Comput. Sci., pages 205–217, Waterloo, Ontario, Canada, Aug 2005.

[DDM00] E.D. Demaine, M.L. Demaine, and J.S.B. Mitchell. Folding Flat Silhouettes and Wrapping Polyhedral Packages: New Results in Computational Origami. Comput. Geom. Theory and Appl., 16(1):3–21, May 2000.

[DMN+10] G. K. Das, A. Mukhopadhyay, S. C. Nandy, S. Patil, and S. V. Rao. Computing the Straight Skeleton of a Monotone Polygon in O(n log n) Time. In Proc. 22nd Canad. Conf. Comput. Geom. (CCCG 2010), pages 207–210, Winnipeg, Canada, Aug 2010.
[DO07] E. D. Demaine and J. O'Rourke. Geometric Folding Algorithms: Linkages, Origami, Polyhedra. Cambridge University Press, Jul 2007.

[EE99] D. Eppstein and J. Erickson. Raising Roofs, Crashing Cycles, and Playing Pool: Applications of a Data Structure for Finding Pairwise Interactions. Discrete Comput. Geom., 22(4):569–592, 1999.

[EGKT08] D. Eppstein, M.T. Goodrich, E. Kim, and R. Tamstorf. Motorcycle Graphs: Canonical Quad Mesh Partitioning. Comput. Graph. Forum, 27(5):1477–1486, Sep 2008.

[Epp00] D. Eppstein. Fast hierarchical clustering and other applications of dynamic closest pairs. J. Exp. Algorithmics, 5, Dec 2000.

[FO99] P. Felkel and Š. Obdržálek. Improvement of Oliva's Algorithm for Surface Reconstruction from Contours. In Proc. 15th Spring Conf. Comput. Graphics, pages 254–263, Budmerice, Slovakia, Apr 1999.

[For00] Steven Fortune. Introduction. Algorithmica, 27(1):1–4, 2000.

[GHR95] R. Greenlaw, H. J. Hoover, and W. L. Ruzzo. Limits to Parallel Computation: PCompleteness Theory. Oxford University Press, Apr 1995.

[GLI] GNU C Library. http://www.gnu.org/software/libc. accessed on May, 2011. [Hav05] S. Havemann. Generative Mesh Modeling. PhD thesis, TU Braunschweig, Braunschweig, Germany, 2005.
[HCK+99] K. Hoff, T. Culver, J. Keyser, M. Lin, and D. Manocha. Fast Computation of Generalized Voronoi Diagrams Using Graphics Hardware. In Comput. Graphics (SIGGRAPH '99 Proc.), pages 277–286, Los Angeles, CA, Aug 1999.

[Hel91] M. Held. On the Computational Geometry of Pocket Machining, volume 500 of Lecture Notes Comput. Sci. Springer-Verlag, June 1991. ISBN 3-540-54103-9.

[Hel94] M. Held. On Computing Voronoi Diagrams of Convex Polyhedra by Means of Wavefront Propagation. In Proc. 6th Canad. Conf. Comput. Geom. (CCCG'94), pages 128–133, Saskatoon, Saskatchewan, Canada, Aug 1994.

[Hel01] M. Held. VRONI: An Engineering Approach to the Reliable and Efficient Computation of Voronoi Diagrams of Points and Line Segments. Comput. Geom. Theory and Appl., 18(2):95–123, Mar 2001.

[HH09a] M. Held and S. Huber. Topology-Oriented Incremental Computation of Voronoi Diagrams of Circular Arcs and Straight-Line Segments. Comput. Aided Design, 41(5):327–338, May 2009.

[HH09b] S. Huber and M. Held. A Practice-Minded Approach to Computing Motorcycle Graphs. In Proc. 25th Europ. Workshop Comput. Geom., pages 305–308, Brussels, Belgium, Mar 2009.

[HH10a] S. Huber and M. Held. Computing Straight Skeletons of Planar Straight-Line Graphs Based on Motorcycle Graphs. In Proc. 22nd Canad. Conf. Comput. Geom. (CCCG 2010), pages 187–190, Winnipeg, Canada, Aug 2010.

[HH10b] S. Huber and M. Held. Straight Skeletons and their Relation to Triangulations. In Proc. 26th Europ. Workshop Comput. Geom., pages 189–192, Dortmund, Germany, Mar 2010.

[HH11a] S. Huber and M. Held. Approximating a Motorcycle Graph by a Straight Skeleton. 2011. (submitted for publication).

[HH11b] S. Huber and M. Held. Motorcycle Graphs: Stochastic Properties Motivate an Efficient Yet Simple Implementation. J. Exp. Algorithmics, 2011. (in press).

[HH11c] S. Huber and M. Held. Theoretical and Practical Results on Straight Skeletons of Planar Straight-Line Graphs. In Proc. 27th Annu. ACM Sympos. Comput. Geom., Paris, France, to be published 2011.

[HLA94] M. Held, G. Lukács, and L. Andor. Pocket Machining Based on ContourParallel Tool Paths Generated by Means of Proximity Maps. Comput. Aided Design, 26(3):189–203, Mar 1994.

[HP00] S. Har-Peled. Constructing Planar Cuttings in Theory and Practice. SIAM J. Comput., 29(6):2016–2039, 2000.

[HS08] J.-H. Haunert and M. Sester. Area Collapse and Road Centerlines based on Straight Skeletons. GeoInformatica, 12:169–191, 2008.

[HS09] M. Held and C. Spielberger. A Smooth Spiral Tool Path for High Speed Machining of 2D Pockets. Comput. Aided Design, 41(7):539–550, July 2009.

[Ink] Inkscape. http://inkscape.org/. accessed on May, 2011.

[IST09] M. Ishaque, B. Speckmann, and C.D. Tóth. Shooting Permanent Rays among Disjoint Polygons in the Plane. In Proc. 25th Annu. ACM Sympos. Comput. Geom., pages 51–60, Aarhus, Denmark, 2009.

[Kle89] R. Klein. Concrete and Abstract Voronoi Diagrams, volume 400 of Lecture Notes in Computer Science. Springer-Verlag, 1989. ISBN 3-540-52055-4.

[KLN09] R. Klein, E. Langetepe, and Z. Nilforoushan. Abstract Voronoi diagrams revisited. Comput. Geom. Theory and Appl., 42(9):885 – 902, 2009.

[KMM93] R. Klein, K. Mehlhorn, and S. Meiser. Randomized Incremental Construction of Abstract Voronoi Diagrams. Comput. Geom. Theory and Appl., 3(3):157–184, 1993.

[KW11] T. Kelly and P. Wonka. Interactive Architectural Modeling with Procedural Extrusions. ACM Trans. Graph., 2011. accepted, not yet published.

[LD03] R. G. Laycock and A. M. Day. Automatically generating large urban environments based on the footprint data of buildings. In Proceedings of the eighth ACM symposium on Solid modeling and applications, SM '03, pages 346–351, New York, NY, USA, 2003. ACM.

[MPF] The GNU MPFR Library. http://www.mpfr.org/. accessed on May, 2011.

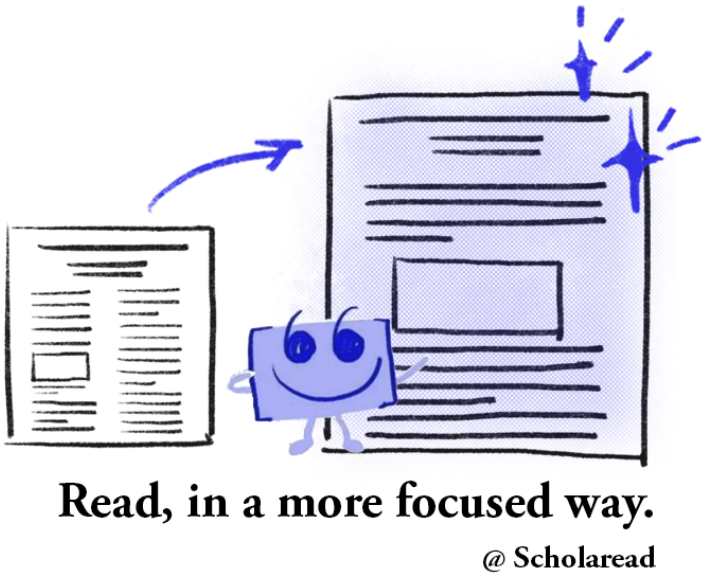
[MWH+06] P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. Van Gool. Procedural Modeling of Buildings. ACM Trans. Graph., 25:614–623, July 2006.

[OPC96] J.M. Oliva, M. Perrin, and S. Coquillart. 3D Reconstruction of Complex Polyhedral Shapes from Contours Using a Simplified Generalized Voronoi Diagram. Comput. Graph. Forum, 15(3):397–408, 1996.

[OSM] OpenStreetMap. http://www.openstreetmap.org. accessed on May, 2011.

[Pap98] E. Papadopoulou. L∞ Voronoi Diagrams and Applications to VLSI Layout and Manufacturing. In Proc. 9th Annu. Internat. Sympos. Algorithms Comput. (ISAAC

Mitered offset for profile machining



[Vya09b] K. Vyatkina. On the Structure of Straight Skeletons. In Transactions on Computational Science VI, volume 5730 of Lecture Notes in Computer Science, pages 362-379. Springer Berlin / Heidelberg, 2009.

[Vya09a] K. Vyatkina. Linear Axis for Planar Straight Line Graphs. In Proc. of thte 15th Australasian Symposium on Computing, volume 94, pages 139-152, Darlinghurst, Australia, 2009. Australian Computer Society.

[TV04a] M. Tanase and R. C. Veltkamp. A Straight Skeleton Approximating the Medial Axis. In Proc. 12th Annu. Europ. Symp. Algorithms (ESA “04), pages 809-821, Bergen, Norway, Sep 2004.

[Yap04] C. K. Yap. Robust Geometric Computation. In J. E. Goodman and J. O'Rourke, editors, Handbook of Discrete and Conputational Geometry, chapter 41, pages 927- 952. Chapmen & Hall/CRC, Boca Raton, FL, 2 edition, 2004.

[TV04b] M. Tanase and R.C. Veltkamp.S Straight Line Skeleton in Linear Time, Topologically Equivalent to the Medial Axis. In Proc. 20th Europ. Workshop Couput. Geom., Mar 2004.

[PC03] S.C. Park and Y.C. Chung. Mitered offset for profile machining. Comput. Aided Design, 35(5):501-505, Apr 2003.

[Sha94] M. Sharir. Almost Tight Upper Bounds for Lower Envelopes in Higher Dimensions. Discrete Comput. Geom., 12:327-345, 1994.

[Yak04] E. Yakersberg. Morphing Betrween Geometric Shapes Using Straight-Skeleton-Based Interpolation. Msc thesis, Technion - Israel Institue of Technology, May 2004.

[Yap87] C.K. Yap. An O(# log n) Algorithm for the Voronoi Diagram of a Set of Simple Curve Segments. Discrete Conput. Geom., 2(4):365-393, 1987.

[TA09] M. Tanase-Avatavului. Shape Decomposition and Retrieoal. PhD thesis, Universiteit Utrecht, Faculteit Wiskunde en Informatica, Utrecht, Netherlands, 2009.

'98), pages 9-18, London, UK, 1998. Springer-Verlag.