Straight Skeletons for General Polygonal Figures in the Plane

OSWIN AICHHOLZER
FRANZ AURENHAMMER
Institute for Theoretical Computer Science
Graz University of Technology
Klosterwiesgasse 32/2, A-8010 Graz, Austria
{oaich,auren}@igi.tu-graz.ac.at

Abstract: A novel type of skeleton for general polygonal figures, the straight skeleton $S(G)$ of a planar straight line graph G, is introduced and discussed. Exact bounds on the size of $S(G)$ are derived. The straight line structure of $S(G)$ and its lower combinatorial complexity may make $S(G)$ preferable to the widely used Voronoi diagram (or medial axis) of G in several applications. We explain why $S(G)$ has no Voronoi diagram based interpretation and why standard construction techniques fail to work. A simple $O(n)$ space algorithm for constructing $S(G)$ is proposed. The worst-case running time is $O(n^3 \log n)$, but the algorithm can be expected to be practically efficient, and it is easy to implement.

We also show that the concept of $S(G)$ is flexible enough to allow an individual weighting of the edges and vertices of G, without changes in the maximal size of $S(G)$, or in the method of construction. Apart from offering an alternative to Voronoi-type skeletons, these generalizations of $S(G)$ have applications to the reconstruction of a geographical terrain from a given river map, and to the construction of a polygonal roof above a given layout of ground walls.

1 Introduction

About twenty years ago, Voronoi diagrams were introduced to theoretical computer science in an influencial paper by Shamos and Hoey [23]. Meanwhile, the Voronoi diagram has become so ubiquitous in geometric algorithms design that some people date the birth of computational geometry to this event. Indeed, a good percentage of papers in the computational geometry literature is directly or indirectly concerned with Voronoi diagrams and their related structures.

We do not attempt here to survey the role of Voronoi diagrams within computational geometry or within related, more practically oriented areas like computer graphics, geometric modeling, robotics, pattern recognition, and geographic information systems. Several survey papers in this spirit are available (by Fortune [8], Aurenhammer [3], and Aurenhammer and Klein [4]), and a recent textbook on this topic exists (by Okabe, Boots, and Sugihara [20]). Instead, we take the opportunity to draw attention to a structure that resembles a Voronoi diagram in many features, but still cannot be computed using the well-developed machinery for Voronoi diagrams.

Voronoi diagram definitions. Speaking sloppily, a Voronoi diagram is a partition of a space U induced by a set S of objects that live in that space. Though the scope of variations of Voronoi diagrams that have been investigated within and outside computational geometry is vast, they all fit into either framework of definition below.

In the *distance model*, a distance function d is defined that maps each element in $S \times U$ to a real number. The Voronoi region of an object $s \in S$ is the set of all elements $x \in U$ whose unique closest object with respect to d is s. Equivalently, the Voronoi diagram of S and d can be expressed as the (projected) lower envelope of the graph of functions $d_s(x)$ over the domain U.

The *wavefront model* prescribes for each object $s \in S$ a set of wavefronts that emanate from s and eventially cover the whole space U. Wavefront propagation stops wherever two wavefronts collide. The Voronoi region of s is the portion of U covered by the wavefronts for s.

In the classical case of a Voronoi diagram, U is the Euclidean plane, S is a finite set of points, and d is the Euclidean distance function. The graph of $d_s(x)$ is a cone which, by applying a suitable transformation, can be mapped into a plane in 3-space such that the projected lower envelope of the planes coincides with the projected lower envelope of the cones. The wavefronts for each point $s \in S$ are circles centered at s. The distance model and the wavefront model are not equivalent in general, however. The structure we are going to investigate in the present paper will have no interpretation in the distance model.

Construction algorithms. Most known efficient construction algorithms for Voronoi diagrams are based on the distance model, or on appropriate lower envelope interpretations thereof. They are based on general algorithmic paradigms like divide-and-conquer or incremental insertion, or on computational geometry-specific techniques like the plane-sweep method. (The documents cited above give extensive lists of relevant references.) The wavefront model, though inherently algorithmic, leads to a construction time complexity at least quadratic in the number of involved objects if applied without the support from sophisticated data structures.

Skeletons. A *planar straight line graph* on n points in the Euclidean plane is a set of non-crossing line segments spanned by these points. In a general setting, a *skeleton* of a planar straight line graph G is a partition of the plane into regions, such that the regions reflect the geometric shape of G in an appropriate manner. We will use the terms *arcs* and *nodes* for denoting the objects that form the boundaries of skeleton regions, in order to distiguish them from the objects forming G, which will be called *edges* and *vertices*.

The well-known and widely used examples of skeletons are the (Euclidean closest-point) Voronoi diagram of G, or if G is a simple polygon, the *medial axis* of G. This skeleton consists of all points in the plane (polygon, respectively) which have more than one closest object in G. Skeletons have numerous applications, for example in biology, geography, pattern recognition, robotics, and computer graphics; see e.g. Kirkpatrick [12], Lee [15], Montanari [18, 19], or Yap [25] for short histories. The Voronoi diagram of G typically contains curved arcs in the neighborhood of the vertices of G. In comparison to the classical Voronoi diagram of n points in the plane, which is composed of straight line segments, this yields disadvantages in the computer representation and construction, and possibly also in the application, of this type of skeleton.

Paper outline. In the present paper, a novel type of skeleton, the *straight skeleton* of a planar straight line graph G, is introduced and discussed. The straight skeleton is obtained as the interference pattern of certain wavefronts propagated from the edges of G. Its arcs are pieces of angular bisectors of the edges of G and thus are line segments themselves. If G is a single *convex* polygon, the part of the straight skeleton interior to G coincides with the medial axis of G. The combinatorial complexity of the straight skeleton is linear in the number of vertices of G, and generally is even less than the complexity of the Voronoi diagram of G.

There have been several attempts to linearize and simplify the Voronoi diagram of planar straight line graphs, mainly for the sake of efficient point location and motion planning; see e.g. Canny and Donald [5], Kao and Mount [11], and McAllister et al. [17]. The so-called compact Voronoi diagram for convex polygons in [17] is particularly suited to these applications as its complexity is linear in the number of polygons rather than in the number of edges. However, its regions do not reflect much of the shape of the polygons which might restrict its applications when being used as a skeleton for polygonal figures.

The straight skeleton comprises the shape of the underlying planar straight line graph G in a natural manner. Moreover, G can be reconstructed easily provided skeleton nodes have been labeled by their distance to G. This fact is considered important in the application of skeletons, for example in picture processing; see Pfaltz and Rosenfeld [21]. We recently learned that the straight skeleton of a set of squares has been proposed as model for polycrystalline growth some thirty years ago by Van der Drift, as an alternative to the Huygens model that just is the Voronoi diagram of the polygons; see Thijssen at al. [24].

Beside its use as a skeleton, we will describe two applications that come from a spatial interpretation of straight skeletons. One is in architecture and concerns the question of constructing a roof rising over a general polygonal outline of walls. Even when the slopes of the roof faces are prescribed, a roof is a highly ambiguous object; see Aichholzer et al. [1]. The straight skeleton provides a canonical solution for this non-trivial task. This has been observed independently in Recuaero and Guti'errez [22]. The other application, in geographic information systems, is the reconstruction of a geographical terrain from a given map that delineates coasts, lakes, and rivers. Terrain slopes and river heights can be chosen individually according to practical requirements. Recently, straight skeletons of polygons have also been applied to origami constructions by Lang [14].

Unlike the Voronoi diagram of G, the straight skeleton of G has no meaningful interpretation in the distance model. That is, it does not correspond to the lower envelope of bivariate functions such that each function is defined only by means of a single edge of G. Even worse, the straight skeleton is no abstract Voronoi diagram in the sense of Klein [13]; it cannot be defined by fixing a separating curve for each pair of edges of G. As a consequence, the well-developed machinery for constructing planar Voronoi diagrams does not apply. We propose a different construction method which is based on simulating the propagation of wavefronts emanating from G. The resulting algorithm is conceptually simple and easy to implement. The only data structures it uses are a triangulation and a priority queue. An upper bound on its running time is $O(n^3 \log n)$, but it shows an $O(n \log n)$ behaviour in typical practical applications. As a byproduct, the algorithm yields an exact upper bound on the number of nodes of a straight skeleton. Very recently, Eppstein and Erickson [6] succeeded in speeding up this algorithm and obtained a subquadratic worst case running time by using tailor-made data structures.

The present paper is an extended version of the conference paper [2].

2 Basic properties of straight skeletons

The definition of the straight skeleton of a planar straight line graph G is based on the connected components of G which we will call *figures* of G. Simple examples of figures are line segments, polygonal lines, or simple polygons. Note that the definition of G excludes single points from being figures. (If appropriate, single points may be modeled by small line segments.) The vertices of G of degree one will play a special role; they are called *terminals* of G.

Imagine each figure F of G as being surrounded by a belt of (infinitesimally small) width ε. For instance, a figure consisting of a single edge e gives rise to a rectangle of length $|e| + 2\varepsilon$ and width 2ε, and a simple polygon gives rise to two homotetic copies of the polygon with minimum distance 2ε. In general, if F partitions the plane into c connected faces then F gives rise to c simple polygons called *wavefronts* of F; see Figure 1. Note that, in this way, each edge of G yields two wavefront edges, whereas each terminal of G yields one wavefront edge.

The wavefronts arising from all the figures of G are now propagated simultanously, at the same speed, and in a self-parallel manner. Wavefront vertices move on angular bisectors of wavefront edges which, in turn, may increase or decrease in length during the propagation. This situation continues as long as wavefronts do not change combinatorially. Basically, there are two types of changes. See Figure 2.

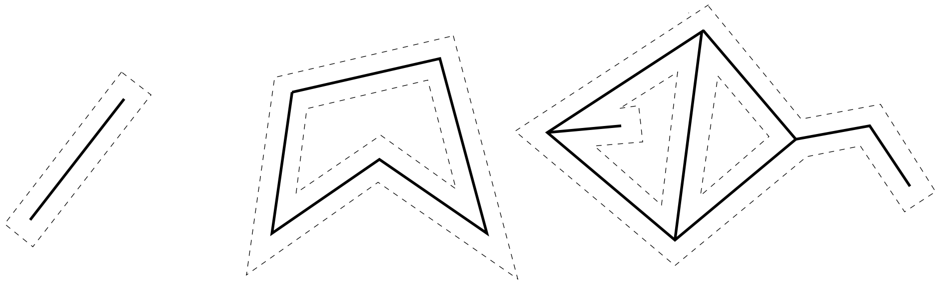

Figure 1: Initial wavefronts

(1) *Edge event*: A wavefront edge collapses to length zero. If its neighboring edges still have positive length then they become adjacent now. The wavefront vanishes, otherwise.

(2) *Split event*: A wavefront edge splits due to interference or self-interference. In the former case, two wavefronts merge into one, whereas a wavefront splits into two in the latter case. New adjacencies occur between the split edge and the wavefront edges that interfered with it.

After either type of event, we are left with a new set of wavefronts which are propagated recursively.

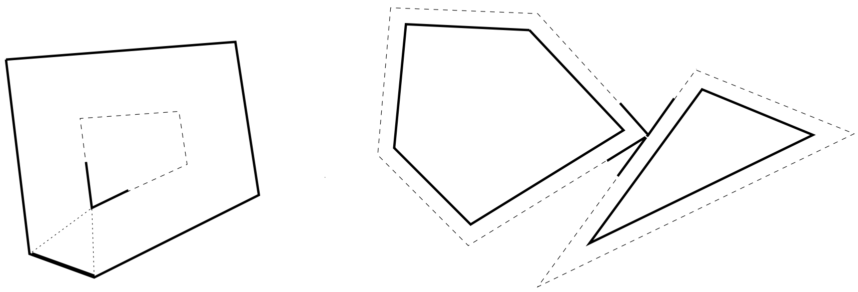

Figure 2: Edge event and split event

The *straight skeleton*, $S(G)$, of G is now defined as the union of the pieces of angular bisectors traced out by wavefront vertices. These bisector pieces are called *arcs*, and their endpoints which are no vertices of G are called *nodes* of $S(G)$. It is important to note that each node corresponds to an edge event or to a split event. $S(G)$ is a unique structure defining a polygonal partition of the plane; see Figure 3.

During the propagation, each wavefront edge e sweeps across a certain area which we call the *face* of e. Each edge of G gives rise to two wavefront edges and thus to two faces, one on each side of the edge. Each terminal of G gives rise to one face. The union of all the faces for a particular figure F of G is called the *region* of F.

Lemma 1 *The faces of $S(G)$ are monotone polygons.*

Proof. The construction of a face f starts with a connected part of f, either at an edge or at a vertex of G. An edge event or a split event cannot disconnect f, even if its defining wavefront edge e is split into several parts. Edge e either propagates to infinity or all of its parts vanish due to edge events. When having vanished, e cannot reappear again. This shows that f is a connected set.

To show monotonicity, let f be a face starting at an edge e of G. (If f starts at a terminal vertex of G, the proof is similar.) We claim that f is monotone in direction e, that is, the intersection of f with every line L normal to e is connected. To get a contradiction, assume that L can be chosen to leave f at some point x and to re-enter f at some point y further from e. Between x and y, L is

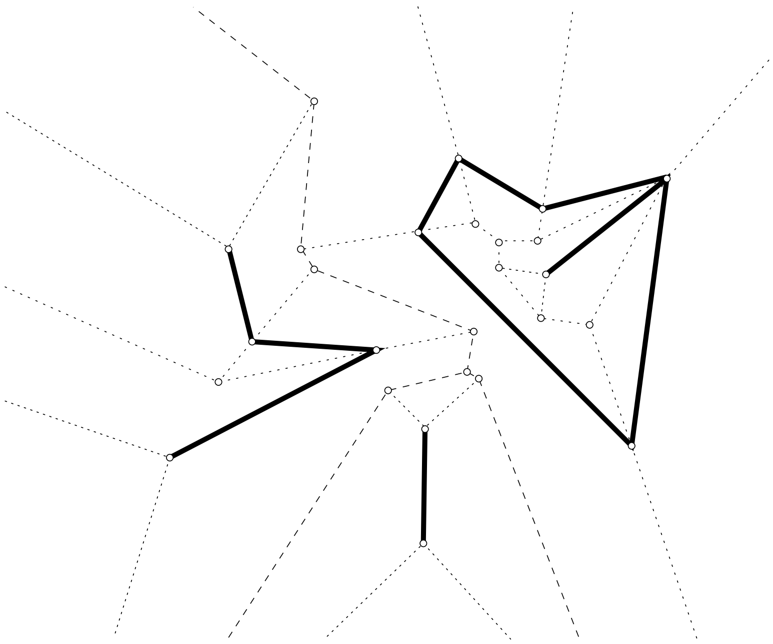

Figure 3: Straight skeleton for three figures

crossed by wavefront edges which are no longer normal to L. Hence they will reach y earlier than do the wavefront edges propagating from e. This contradicts the definition of y and completes the proof. $\square$

Lemma 1 implies that $S(G)$ partitions the plane into exactly $2m + t = O(n)$ simply connected faces, if G consists of m edges, t terminals, and n vertices. As $S(G)$ is a planar graph with node degree of at least three, the number of arcs and nodes of $S(G)$ is $O(n)$, too. Below we state an exact bound on the number of nodes of $S(G)$. The bound includes one node at infinity for each unbounded arc of $S(G)$. The proof is postponed to Section 3.

Lemma 2 *Let G be a planar straight line graph with t terminals and totally n vertices. The number of (finite and infinite) nodes of $S(G)$ is exactly $2n + t - 2$.*

The corresponding exact bound for the *Voronoi diagram* of G is larger in general, namely $2n + t + r - 2$, where r counts the reflex angles ($\pi < \alpha < 2\pi$) formed by G; see Aurenhammer and Klein [4]. Interestingly, the number of edges and figures of G is irrelevant in both cases. Both bounds are maximum, $3n - 2$, when G consists of $\frac{n}{2}$ disjoint line segments. If G is a simple polygon then the part of $S(G)$ interior to G forms a tree whose leaves are the n vertices of the polygon. Thus there are only $n - 2$ nodes, whereas the medial axis of G has $n + r - 2$ nodes if there are r reflex interior angles; see Lee [15]. An upper bound of $4n - 3$ on the number of nodes of the Voronoi diagram of G has been proved earlier, by Lee and Drysdale [16].

The wavefront model yielding $S(G)$ is similar to the wavefront model that yields the Voronoi diagram of G. Some comments are in order to point out the differences between both models.

In the Voronoi diagram model, all points on a wavefront for a figure F have the same *distance* to F. Therefore wavefronts are not polygonal in general but contain circular arcs. In the straight skeleton model, all points on a wavefront have the same *offset* from F. So all wavefronts are polygonal, and a wavefront vertex moves away from F faster than its neighborhood. Speed is controlled by the angle spanned by the wavefront edges incident to the vertex. This may make $S(G)$ behave completely different from the Voronoi diagram of G, in a geometric and combinatorial sense.

It is desirable to find a non-procedural definition of $S(G)$, as it is available for the Voronoi diagram of G by measuring distances from G. The obvious approach is to extract a distance function from the wavefront model. Let x be a point in the plane and let F be a figure of G. There is a unique wavefront W for F that passes through x. The minimum distance between W and F is taken as the distance $d(x, F)$ between point x and figure F.

To see what happens when using this distance function, let us express $d(x, F)$ by the bivariate function $\varphi_F(x) = d(x, F)$. The Voronoi diagram of G under the distance function d then corresponds to the lower envelope of the functions φ_F for all figures F of G. Lower envelope interpretations of Voronoi diagrams have been used systematically in Edelsbrunner and Seidel [7].

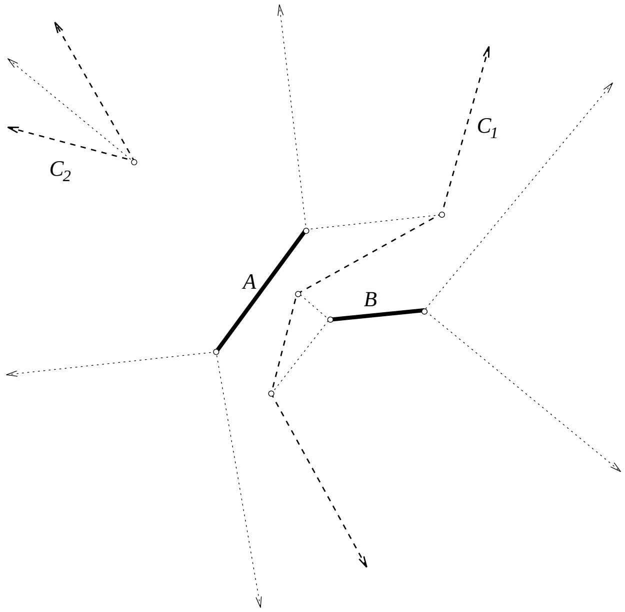

Figure 4: Segments A and B have two separating curves

Figure 4 displays the Voronoi diagram for the distance function d and two single-edge figures A and B. The contribution of φ_B is disconnected, and it is separated from that of φ_A by two polygonal curves C_1 and C_2. However, in the straight skeleton for A and B, curve C_2 does not appear, as the propagation of wavefronts ceases at points of interference. This reflects a significant difference between the two structures: in $S(G)$, the domain of influence of $d(x, B)$ depends on the location of other figures. We conclude that, without prior knowledge of its regions, $S(G)$ cannot be defined by means of distances from the figures.

It is tempting to try to exclude unintended separating curves (C_2 in Figure 4) by simply defining the separator of two edges as the interference pattern of their wavefronts (C_1 in Figure 4). However, $S(G)$ fails to be the abstract Voronoi diagram (see Klein [13]) that results from the separators for all pairs of edges of G. The main problem with this approach is that a point common to the separators of figures A and B, and A and C, respectively, need not belong to the separator of B and C. This causes the existence of 'no-mans lands' that contain points belonging to neither region.

3 A simple skeleton construction algorithm

We now turn to the problem of computing the straight skeleton $S(G)$ for a given planar straight line graph G.

We have argued in the preceding section that $S(G)$ has no Voronoi-type structure. This undesirable property rules out the application of standard Voronoi diagram construction methods. In particular, powerful techniques like incremental insertion, and with it, divide-and-conquer fail to work.

As the straight skeleton is well defined for arbitrary subgraphs of G, it is instructive to recall why incremental insertion of its edges is still doomed to fail as a construction method. Insertion of a new edge e does not only involve the creation of new faces by propagating e. A prior presence of e possibly would have altered the propagation of wavefronts for figures which are not neighbored to e after its propagation. In other words, parts of the skeleton exterior to the newly inserted region for e may have to be deleted and reconstructed.

The construction of the regions of $S(G)$ thus has to be carried out simultaneously. The algorithm to be described now is an implementation of the wavefront definition of $S(G)$.

Basically the algorithm keeps, throughout the propagation, a triangulation of the part of the plane that has not been reached yet by some wavefront. The vertices of this triangulation are just the vertices of the current wavefronts. They move on angular bisectors as the propagation proceeds, and triangles will change their shape and will collapse under certain circumstances. The crucial point is that each edge event and each split event for a wavefront will be witnessed by a collapsing triangle. Triangles are held in a priority queue which is structured by collapsing time.

Let us look into the details of this simple algorithm. In a first step, the initial wavefronts are generated for each figure of G by duplicating its vertices and linking them accordingly; cf. Figure 1. Then the vertex set of G is triangulated in an arbitrary manner. The newly introduced triangulation edges are called *spokes*, to avoid confusion with the edges of wavefronts or figures. Spokes have to be assigned carefully to duplicates of figure vertices such that — immediately after the propagation of wavefronts has got started — the area swept over is untriangulated, and its complement is triangulated.

Lemma 3 *Let G have n vertices, t of which are terminals. The initial triangulation of the vertices of the wavefronts for G has exactly $2n + t - 2$ (bounded and unbounded) triangles.*

Proof. Each vertex v of G of degree $d \geq 2$ is duplicated into d wavefront vertices. Spokes (and triangles) incident to v are shared among these vertices as determined by the edges of G incident to v. Each terminal u of G is duplicated into two vertices which are linked by a wavefront edge e. One copy keeps all the spokes incident to u. The second copy gets assigned only one spoke, which is new and partitions the quadrilateral based on e into two triangles. In this way, a new triangle is created which has not been incident to u before.

When triangulating the n vertices of G, we partition the plane into exactly $2n - 2$ bounded or unbounded triangles. These are shared among the wavefront vertices. In addition, one new triangle is created for each terminal of G. This implies the claimed number $2n + t - 2$ of triangles in the initial triangulation. □

The topology of the triangulation changes whenever the vertices of a triangle get collinear during the propagation. Such a collapse of triangles arises in three different ways. Let v be a wavefront vertex and see Figure 5.

(1) *Flip event*: v sweeps across a spoke s. To keep things triangulated, we remove s and insert the spoke t.

(2) *Edge event*: v merges with another vertex of the wavefront, which has just lost an edge e. We update the triangulation by identifying these two vertices and removing e.

(3) *Split event*: v hits a wavefront edge e by splitting it into two edges e' and e''. We duplicate v, assign e' and e'' and the formerly incident spokes of v to these vertices accordingly, and remove e.

At each edge event or split event, a new node of $S(G)$ is produced. The algorithm terminates when the collapsing time of all triangles in the priority queue is infinite. By using an inductive argument, the correctness of the algorithm can be proved easily.

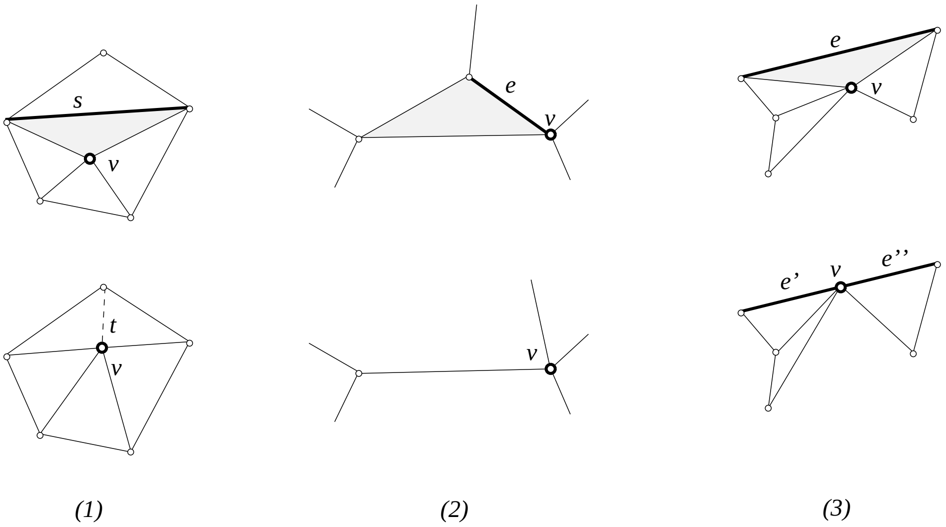

Figure 5: Flip event, edge event, and split event

Lemma 4 *Let G have n vertices and t terminals. The total number of edge events and split events is bounded by $2n + t - 2$.*

Proof. The argument is based on counting the number of triangles in the triangulation maintained by the algorithm. By Lemma 3, there are $2n + t - 2$ triangles at the beginning. Each flip event obviously leaves the actual number of triangles unchanged. Moreover, each edge event and each split event decrease this number by exactly one. The claimed upper bound follows immediately. □

As each node of $S(G)$ is created either by an edge event or by a split event, the total number of nodes obeys the same bound. The bound is exact when we also count the infinite nodes at unbounded arcs of $S(G)$ (which also have to be stored in some way in the data structure representing $S(G)$). After the very last event, the remaining triangles are all unbounded. These are the triangles with infinite collapsing time. Their unbounded spokes correspond to the infinite nodes of $S(G)$. This gives a proof of Lemma 2 in Section 2.

Another consequence of Lemma 3 is that, at each point in time, at most $2n + t - 2$ triangles have to be stored. The storage requirement of the algorithm thus is $O(n)$.

For the analysis of the runtime, we also need to bound the number of flip events. To this end, we distiguish between *convex* and *reflex* vertices of a wavefront, depending on whether or not the wavefront is locally convex at the vertex as seen from the unswept area. A convex vertex can never sweep across a spoke, as the spoke would then intersect an area which already has been swept over by the wavefront and thus is untriangulated. This implies that flip events are caused only by reflex vertices. We make another simple observation.

Observation 1 *Each reflex wavefront vertex disappears after the first non-flip event it causes.*

As a consequence, each of the $O(n)$ reflex wavefront vertices that are present at the beginning moves on a straight line until it eventially disappears at an edge event — where it merges with a convex vertex — or a split event — where it splits into two convex vertices. Three points that move linearly and at constant but individual speed can become collinear at most twice. So each triple of reflex wavefront vertices gives rise to at most two flip events.

Lemma 5 *The total number of flip events is $O(n^3)$.*

By Lemmas 4 and 5, the total number of triangles processed is $O(n^3)$. Apart from updating the priority queue holding these triangles, updates concerning the speed of the wavefront vertices that span these triangles have to be performed. Clearly, a vertex does not change its speed at a flip event. Each edge event and each split event, however, causes a change in the amount and direction of speed for the involved vertex v. This change alters the collapsing time of all triangles that currently have v at a vertex. We have to recompute these collapsing times and restructure the priority queue accordingly. As the degree of a vertex is bounded by $O(n)$, Lemma 4 implies that, in total, $O(n^2)$ triangles have their collapsing times updated.

We thus get an $O(n^3 \log n)$ time and $O(n)$ space algorithm for computing straight skeletons. The upper bound on the running is even worse than for the trivial method, which checks each wavefront vertex against each wavefront edge for computing the next event, and runs in $\Theta(n^3)$ time. However, the poor worst-case behaviour does not seem to be a serious drawback of our algorithm in practical applications. For example, for typical input graphs arising from the applications described in the next section, we observed a running time close to $O(n \log n)$. In fact, the time for constructing $S(G)$ did not significantly exceed the time for computing the initial triangulation for G.

When the initial triangulation is available, the basic step of the algorithm is a collinearity test for three points moving constantly on straight lines. The test amounts to the resolution of a quadratic equation in one variable — the collapsing time of the triangle spanned by the points.

4 Roofs and terrains

Beside its use as a skeleton for polygonal figures, the straight skeleton has interesting applications that come from a three-dimensional interpretation.

Let G and $S(G)$ be a planar straight line graph and its straight skeleton, respectively. $S(G)$ induces a distance function T with respect to G. Namely, given a point x in the plane, $T(x, G)$ just is the unique time when x is reached by the first wavefront edge. The face of $S(G)$ containing x indicates the edge or terminal of G that sends out the corresponding wavefront edge. In Section 2 we have argued why T cannot be defined locally by means of the figures of G, but rather has to require knowledge of the faces of $S(G)$. Note that G is the zero set of T.

Now consider the function $\Sigma_G(x) = T(x, G)$ on the plane. It is easy to see that Σ_G is continuous and piece-wise linear, that is, its graph is a polygonal surface in three-space. Its facets project vertically to the faces of $S(G)$, and its intersection with the plane gives G. Below we mention two applications where the construction of a surface from a given planar straight line graph G comes in.

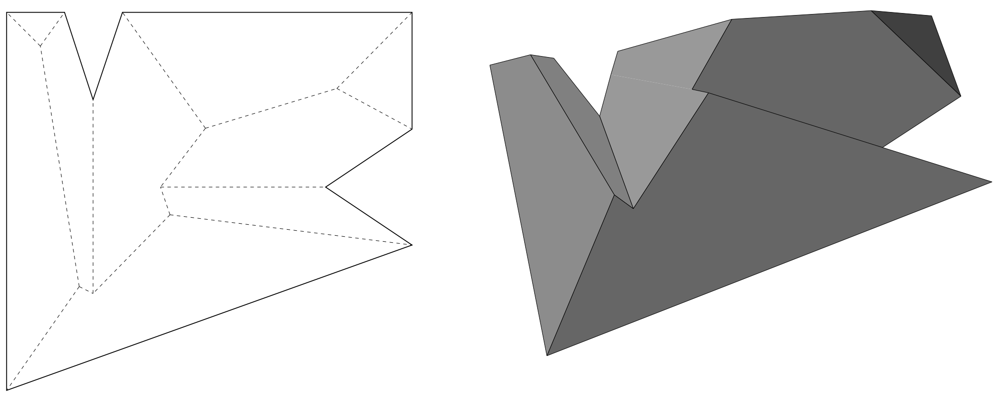

Figure 6: Straight skeleton and corresponding roof

Let G be a simple polygon, interpreted as an outline of a building's groundwalls. The task is to construct a polygonal roof that rises over G and whose roof facets are of the same slope.

If G is a rectilinear (and axis-aligned) polygon then the medial axis for G under the L_∞-metric gives a solution. Actually, $S(G)$ coincides with this structure in that case. The usual Euclidean medial axis is not suited even in this special case, as it gives rise to cylindrical roof facets.

For general shapes of G, the construction of a roof is by no means trivial: a roof, defined as a polygonal surface with given facet slopes and given intersection with the plane, is an highly ambigous object; see Aichholzer et al. [1]. The surface Σ_G, when restricted to the interior of G, constitutes a canonical and general solution. A construction of this type has also been used independently by Recuaero and Guti'errez [22]. Figure 6 gives an illustration. Because of the tree structure of $S(G)$ in the interior of G, the roof obtained from Σ_G has exactly $n-2$ nodes and $2n-3$ arcs. This is minimum for all possible roofs of an n-gon G.

In this context, two generalizations of $S(G)$ are appropriate. In the surface Σ_G as defined above, all facets have the same slope. However, the concept of straight skeleton is flexible enough to be adapted to yield surfaces (and in particular, roofs) with prescribed facet slopes. This is achieved by tuning the propagation speed of the individual wavefront edges. Of course, this changes the geometric and topological structure of the skeleton. Its faces, though remaining connected, need not be monotone or simply connected. However, the upper bound on the skeleton size in Lemma 2, and the construction algorithm of Section 3 remain valid.

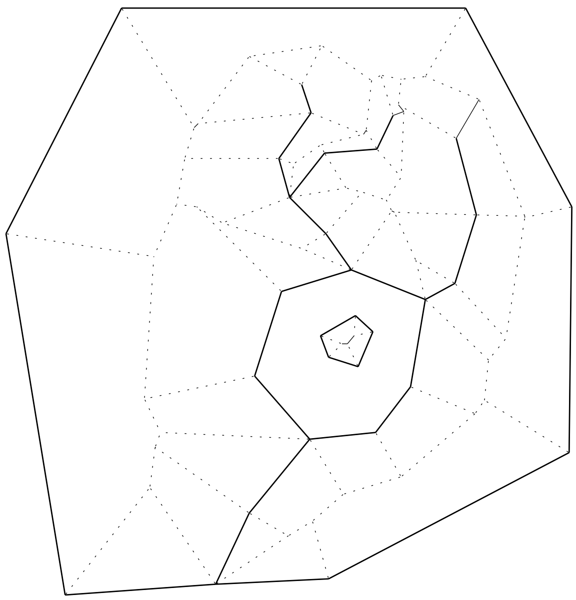

Figure 7: Coastline and river map (solid lines)

To exploit the concept to its utmost generality, individual heights for the surface points that correspond to vertices of G may be specified in addition. To deal with this situation, wavefronts are not propagated parallel to G's edges but at a certain angle and offset that is determined by the relative heights of the vertices. The upper bound in Lemma 2 and the construction algorithm still remain valid. The only requirement needed for a proper definition of the skeleton is that speeds and angles of wavefront edges are chosen such that each point in the plane is reached by wavefronts at only one point in time.

These generalizations of $S(G)$ are similar to the concepts of multiplicatively and additively weighting of Voronoi diagrams; see e.g. Aurenhammer [3]. Unlike straight skeletons, however, weighted Voronoi diagrams may exhibit a completely different behavior than their unweighted counterparts. For instance, regions in the multiplicatively weighted Voronoi diagram for points are disconnected in general.

Another interesting application, which makes use of the general shape of the underlying graph G, is the reconstruction of geographical terrains. Assume we are given a map where rivers, lakes, and coasts are delineated by polygonal lines, yielding a planar straight line graph G; see Figure 7. We are requested to reconstruct a corresponding polygonal terrain from G, possibly with additional information concerning the elevation of lakes and rivers, and concerning the slopes of the terrain according to different mineralogical types of material. The surfaces resulting from $S(G)$ and its modifications seem to meet these general geographical requirements in an appropriate manner. Figure 8 gives an example.

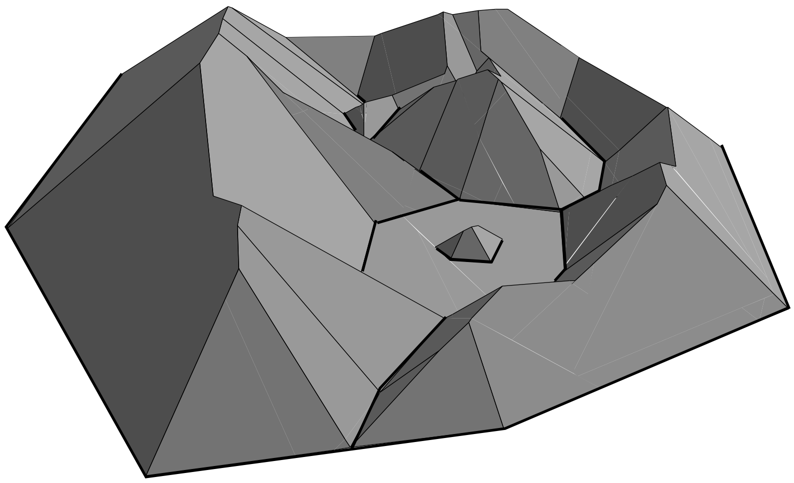

Figure 8: Terrain reconstructed from the map in Figure 7

A related question is the study of rain water fall and its impact on the floodings caused by rivers in a given geographical area. Currently, the amount of water drained off by a river is estimated by means of the Voronoi diagram of the river map, as reported by Gold [10]. This models the assumption that each raindrop runs off to the river closest to it, which might be unrealistic in certain situations. The straight skeleton offers a more realistic model by bringing the slopes of the terrain into play. In particular, the surface Σ_G that arises from $S(G)$ (in its original unweighted form) has the following nice property: every raindrop that hits a surface facet f runs off to the edge or terminal of G defining f. More formally, let x be a point on Σ_G, and let $g(x)$ denote the path that starts at x and follows the steepest gradient on Σ_G.

Lemma 6 *Let x be a point on a facet f of Σ_G. Then $g(x)$ ends at the unique edge or terminal of G that defines f.*

Proof. By the monotonicity of the faces of $S(G)$ stated in Lemma 1, $g(x)$ reaches the boundary of f exactly once, at point y, say. If $y \in G$ then we are done. Else y lies on an arc, b, of Σ_G that projects to a reflex arc of $S(G)$, i.e. an arc traced out by a reflex wavefront vertex. This is because only reflex arcs form an angle larger than 90 degree with their defining wavefront edges. Clearly, $g(x)$ follows arc b to its lowest point which, by Observation 1, has to be a (terminal or non-terminal) endpoint of an edge of G. $\square$

5 Concluding remarks

We have introduced an alternative type of skeleton for general polygonal figures in the plane, and have discussed some of its properties, applications, and generalizations. The general advantages of the straight skeleton, compared to the Voronoi diagram, are its straight line structure and its lower combinatorial complexity. We believe the straight skeleton to be of use in many practical applications.

In view of the existing $O(n \log n)$ time methods for computing Voronoi diagrams of planar straight line graphs (see Fortune [9], Kirkpatrick [12], Yap [25]), the available construction algorithms for straight skeletons call for improvement in runtime. The recent algorithm by Eppstein and Erickson [6] runs in time $O(n^{1+\varepsilon} + n^{8/11+\varepsilon} k^{9/11+\varepsilon})$, where $k = r + 2t = O(n)$, and r and t count the reflex vertices and terminals of the planar straight line graph, respectively. Note that k is just the total number of reflex vertices in the initial wavefronts. Though theoretically efficient, the algorithm suffers from the use of complicated data structures and may be beaten by the simple method in the present paper for most practical inputs.

The definition of the straight skeleton $S(G)$ can be modified by considering as figures the individual edges of G rather than the connected components. In other words, each edge of G is now assumed to send out its own rectangular wavefront. The resulting structure has more similarities to the Voronoi diagram of G than does $S(G)$, as the speed of wavefront vertices is bounded by a factor of $\sqrt{2}$ with respect to the propagation speed. However, the size of the skeleton increases slightly, as four arcs instead of two emanate from each vertex where G forms an acute angle . Both structures are identical if no acute angles occur in G. Figure 9 illustrates this modification for the interior of a simple polygon.

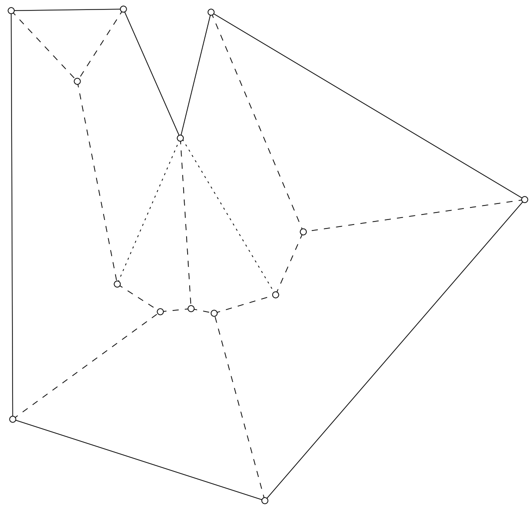

Figure 9: Modified straight skeleton for a polygon

Finally, a generalization of $S(G)$ to higher dimensions is of interest. Applications to efficient motion planning in a 3D polyhedral environment seem possible. The piecewise linearity of $S(G)$ is a crucial advantage in 3D, as the complicated curved surfaces arising in a Voronoi diagram for polyhedral objects restrict its practical use. In particular, the skeleton of a single non-convex polytope constitutes a partition into simpler polytopes which may be useful in the context of solid modeling. We will elaborate on straight skeletons in 3D in a separate paper.

Acknowledgements: The second author would like to express thanks to G.L. Sicherman from AT&T Bell Labs. for drawing his attention to angle bisector skeletons. Discussions on the presented topic with J.-D. Boissonnat, O. Devillers, H. Edelsbrunner, M. Formann, R. Klein, D.T. Lee, F.P. Preparata, G. Rote, and K. Varadarajan are gratefully acknowledged. Thanks also go to T. Natschläger and H. Ramoser for implementing an algorithm for visualizing terrains.

References

[1] O. Aichholzer, D. Alberts, F. Aurenhammer, and B. Gärtner, *A novel type of skeleton for polygons*, J. Universal Comput. Sci. 1 (1995), 752 - 761.

[2] O. Aichholzer and F. Aurenhammer, *Straight skeletons for general polygonal figures*, Proc 2^{nd} Ann. Int'l. Computing and Combinatorics Conf. CoCoon '96, Springer LNCS 1090 (1996), 117 - 126.

[3] F. Aurenhammer, *Voronoi diagrams — a survey of a fundamental geometric data structure*, ACM Computing Surveys 23, 3 (1991), 345 - 405.

[4] F. Aurenhammer and R. Klein, *Voronoi Diagrams*, in: J.R. Sack and G. Urrutia (eds.), Handbook on Computational Geometry, Elsevier, to appear.

[5] J. Canny and B. Donald, *Simplified Voronoi diagrams*, Discrete & Computational Geometry 3 (1988), 219 - 236.

[6] D. Eppstein and J. Erickson, *Raising roofs, crashing cycles, and playing pool: Applications of a data structure for finding pairwise interactions*, Manuscript, Comput. Sci. Dept., Duke Univ., Durham, NC, U.S.A., 1997.

[7] H. Edelsbrunner and R. Seidel, *Voronoi diagrams and arrangements*, Discrete & Computational Geometry 1 (1986), 25 - 44.

[8] S. Fortune, *Voronoi diagrams and Delaunay triangulations*, in: Computing in Euclidean Geometry, D.-Z. Du and F. Hwang (eds.), Lecture Note Series on Computing 4 (1995), 225 - 262.

[9] S. Fortune, *A sweep line algorithm for Voronoi diagrams*, Algorithmica 2 (1987), 153 - 174.

[10] C. Gold, personal communication, 1995.

[11] T.C. Kao and D.M. Mount, *An algorithm for computing compacted Voronoi diagrams defined by convex distance functions*, Proc. 3^{rd} Canadian Conf. Computational Geometry (1991), 104 - 109.

[12] D.G. Kirkpatrick, *Efficient computation of continuous skeletons*, Proc. 20^{th} Ann. IEEE Symp. on FOCS (1979), 18 - 27.

[13] R. Klein, *Concrete and Abstract Voronoi diagrams*, Springer LNCS 400 (1989).

[14] R.J. Lang, *A computational algorithm for origami design*, Proc. 12^{th} Ann. ACM Symp. Computational Geometry (1996), 98 - 105.

[15] D.T. Lee, *Medial axis transformation of a planar shape*, IEEE Trans. Pattern Analysis and Machine Intelligence, PAMI-4 (1982), 363-369.

[16] D.T. Lee and R.L. Drysdale, *Generalization of Voronoi diagrams in the plane*, SIAM J. Computing 10 (1981), 73 - 87.

[17] M. McAllister, D.G. Kirkpatrick, and J. Snoeyink, *A compact piecewise-linear Voronoi diagram for convex sites in the plane*, Discrete & Computational Geometry 15 (1996), 73 - 105.

[18] U. Montanari, *A method for obtaining skeletons using a quasi-Euclidean distance*, J. ACM 15 (1968), 600 - 624.

[19] U. Montanari, *Continuous skeletons from digitized images*, J. ACM 16 (1969), 534 - 549.

[20] A. Okabe, B. Boots, and K. Sugihara, *Spatial Tesselations: Concepts and Applications of Voronoi diagrams*, John Wiley & Sons, Chichester, England, 1992.

[21] J.L. Pfaltz and A. Rosenfeld, *Computer representation of planar regions by their skeletons*, Comm. ACM 10,2 (1967), 119 - 125.

[22] A. Recuaero and J.P. Guti'errez, *Sloped roofs for architectural CAD systems*, Microcomputers in Civil Engineering 8 (1993), 147 - 159.

[23] M.I. Shamos and D. Hoey, *Closest point problems*, Proc. 16^{th} Ann. IEEE Symp. on FOCS (1975), 151 - 162.

[24] J.M. Thijssen, H.J.F. Knops, and A.J. Dammers, *Dynamic scaling in polycrystalline growth*, Manuscript, Physics Dept., Univ. of Nijmegen, The Netherlands, 1991.

[25] C.-K. Yap, *An $O(n \log n)$ algorithm for the Voronoi diagram of a set of simple curve segments*, Discrete & Computational Geometry 2 (1988), 365 - 393.