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

1 Introduction

A *planar straight line graph*, G, on n points in the Euclidean plane is a set of non-crossing line segments spanned by these points. A *skeleton* of G is a partition of the plane into faces that reflect the shape of G in an appropriate manner. The well-known and widely used examples of skeletons are the *medial axis* of a simple polygon or, more generally, the (closest-point) *Voronoi diagram* of G. Skeletons have numerous applications, for example in biology, geography, pattern recognition, robotics, and computer graphics; see e.g. [Ki, L, Y] for a short history.

The Voronoi diagram of G consists of all points in the plane which have more than one closest object in G. Typically, it contains curved arcs in the neighborhood of the vertices of G. This is considered a disadvantage in the computer representation and construction, and sometimes also in the application, of this type of skeleton.

There have been several attempts to linearize and simplify Voronoi diagrams of planar straight line graphs, mainly for the sake of efficient point location and motion planning [CD, KM, MKS]. The compact Voronoi diagram for convex polygons in [MKS] 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 faces do not reflect much of the shape of the polygons which might restrict its application when being used as a skeleton for polygonal figures.

In the present paper, a novel type of skeleton, the *straight skeleton* of G, is introduced and discussed. Its arcs are pieces of angular bisectors of the edges of G. Its combinatorial complexity is in general is even less than the complexity of the Voronoi diagram of G. Still, G can be reconstructed easily from its straight skeleton. This fact is considered important in certain applications of skeletons [PR]. Beside its use as a skeleton, we describe two applications that come from a spatial interpretation of straight skeletons. One concerns the question of constructing a roof above a general polygonal outline of ground walls. The other application is the reconstruction of a geographical terrain from a given map that delineates coasts, lakes, and rivers.

We define the straight skeleton as the interference pattern of certain wavefronts propagated from the edges of its underlying graph G. A different wavefront model (or growth model) is well known to yield the Voronoi diagram of G. The straight skeleton, however, has no Voronoi diagram based interpretation, neither in terms of distances nor as an abstract Voronoi diagram for bisecting curves. As a consequence, the well-developed machinery for constructing planar Voronoi diagrams does not apply. We propose a different construction algorithm, which is conceptually simple and easy to implement. The only data structures it uses are a triangulation and a priority queue. Its worst-case running time is $\Theta(n^2 \log n)$ for special shapes of G, but should be close to $O(n \log n)$ in typical practical applications. As a byproduct, the algorithm enables us to prove an exact bound on the number of nodes in a straight skeleton.

2 Basic properties of straight skeletons

The definition of the straight skeleton of a planar straight line graph G is based on its connected components which will be called the *figures* of G. Note that the definition of G excludes single points from being figures. If appropriate, points may be modeled by small line segments. The vertices of G of degree one will play a special role; they are called *terminals* in the sequel.

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.

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.

(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.

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 $S(G)$. 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 1.

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.

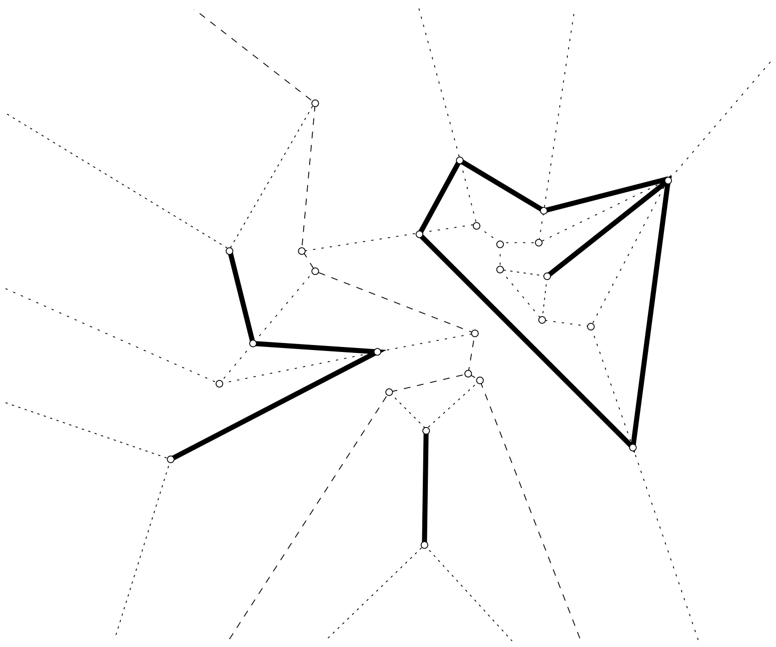
Figure 1: Straight skeleton for three figures

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

Proof. Omitted in this abstract.

Lemma 1 implies that $S(G)$ partitions the plane into exactly $2m + t = O(n)$ simply connected faces, if G realizes m edges, t terminals, and n vertices. As two faces can have at most one arc or one edge in common, 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 $S(G)$ that also includes the nodes at infinity for the unbounded arcs 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 formed by G; see [AK]. 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 has only $n - 2$ nodes, whereas the medial axis of G has $n + r - 2$ nodes if there are r reflex interior angles [L].

The wavefront model yielding $S(G)$ is very similar to the model sometimes used to define 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 minimum distance to F. Therefore wavefronts are not polygonal in general but contain circular arcs. In the straight skeleton model, all wavefronts are polygonal. So a wavefront vertex may move away from F faster than other parts. 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 [ES] of the functions φ_F for all figures F of G.

Figure 2 displays this type of Voronoi diagram for 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 2) by simply defining the separator of two edges as the interference pattern of their wavefronts (C_1 in Figure 2). However, $S(G)$ fails to be the abstract Voronoi diagram [Kl] 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 A and B, and A and C, respectively, need not belong to the separator of B and C.

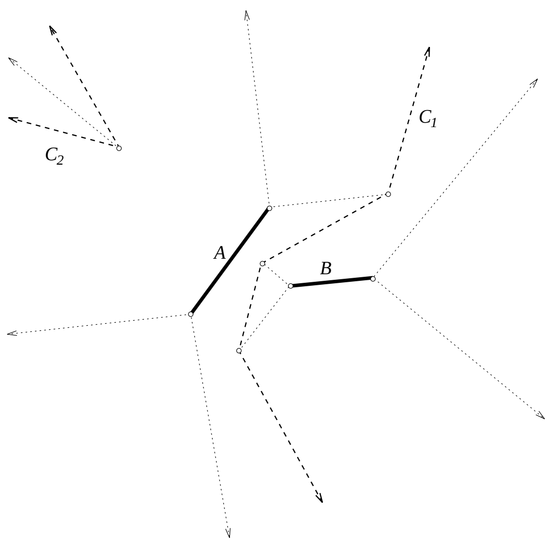

Figure 2: Segments A and B have two separating curves

3 A simple skeleton construction algorithm

The fact that $S(G)$ can neither be defined by using distances nor by using separating curves 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, too.

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.

In a first step, the initial wavefronts are generated for each figure of G by duplicating its vertices and linking them accordingly. 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 get 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 u 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 3.

(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.

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

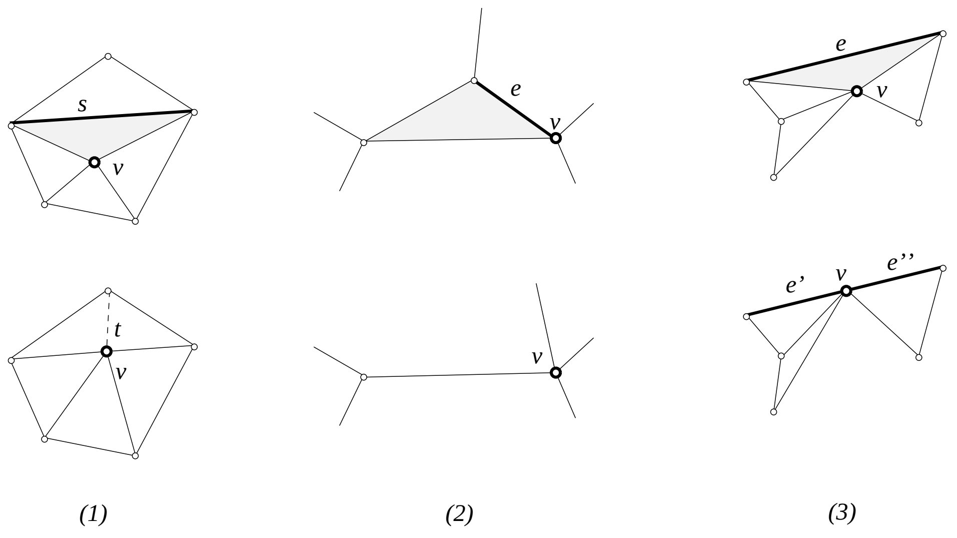

Figure 3: Flip event, edge event, and split event

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 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.

Lemma 5 *The total number of flip events is* $O(n^2)$.

Proof. Omitted in this abstract.

By Lemmas 4 and 5, the total number of triangles processed is $O(n^2)$. 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 $O(n^2 \log n)$ time and $O(n)$ space algorithm for computing straight skeletons. The running time actually is attained for specially constructed input graphs. However, the poor worst-case behaviour does not seem to be a serious drawback of the 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. By means of $S(G)$, a distance function d with respect to G can be defined. Namely, given a point x in the plane, $d(x, G)$ just is the unique time when x is reached by a wavefront. Clearly, $d(x, G) = 0$ for x on G. Now consider the function $\Sigma_G(x) = d(x, G)$ 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.

Let G be a simple polygon, interpreted as an outline of a building's groundwalls. The task is to construct a polygonal roof above G when slopes are given for the roof facets. If G is a rectilinear (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 [AAAG]. The surface Σ_G, when restricted to the interior of G, constitutes a canonical and general solution; see Figure 4 for an example. It is easy to see that the roof obtained from Σ_G has exactly $n - 2$ nodes and $2n - 3$ arcs, which 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 (in particular, roofs) with prescribed facet slopes. This is achieved by tuning the propagation

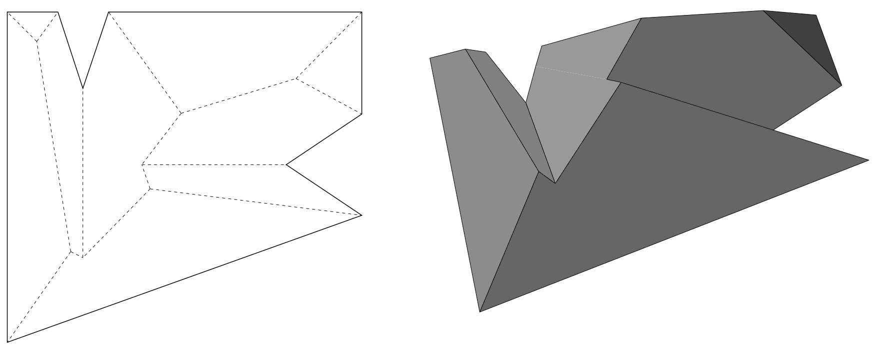
Figure 4: Skeleton and corresponding roof

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 any more. However, the upper bound on the skeleton size in Lemma 2, and the construction algorithm of Section 3 remain valid.

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 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 [A]. 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.

An application that makes use of this general concept of skeleton is the reconstruction of terrains. Assume we are given a map where rivers, lakes, and coasts are delineated by polylines, giving a planar straight line graph G. 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 meet these general geographical requirements in an appropriate manner.

A related question is the study of rain water fall and its impact on the floodings caused by rivers in a given geographic area. Currently, the amount of water drained off by a river is estimated by means of the Voronoi diagram of the river map [G]. 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 fact, we can show that the surface that arises from $S(G)$ (in its original 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.

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 for line segments, 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 $O(n \log n)$ methods [Ki, L, Y] for Voronoi diagrams of planar straight line graphs, the proposed construction algorithm calls for improvement in runtime. It seems possible to gain efficiency in the worst case by maintaining a triangulation of low stabbing number during the wavefront propagation.

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.

Finally, a generalization of $S(G)$ to three dimension 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: We would like to thank Herbert Edelsbrunner for a discussion on the presented algorithm. Preliminary work on this subject was done while the second author visited the Leonardo Fibonacci Institute, Trento, Italy.

References

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

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

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

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

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

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

[KM] 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.

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

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

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

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

[MKS] 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.

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

[Y] 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.