A Note on Generalizations of Chew's Algorithm for the Voronoi Diagram of a Convex Polygon

Rolf Klein
FernUniversität Hagen

Andrzej Lingas
Lund University

Abstract

We present linear-time generalizations of Chew's randomized algorithm for the Voronoi diagram of a convex polygon to include the convex hull of certain polygons in 3-space and the Voronoi diagram of monotone line segment sets in the plane.

1 Introduction

The Voronoi diagram and its dual, the Delaunay triangulation, belong to the most useful structures in computational geometry, see [2] and [12] . Computing the diagram of n points in the Euclidean plane is well known to have $\Theta(n \log n)$ time-complexity. But since the Voronoi diagram is so useful it is natural to ask if faster algorithms are available for special site configurations.

In 1987, Aggarwal, Guibas, Saxe, and Shor [1] derrived a linear-time method for computing the convex hull of polygons in 3-space having a convex projection on a 2-dimensional hyperplane. As a corollary they obtained a linear-time upper bound for the problem of computing the Voronoi diagram of the vertices of a convex polygon. They also claimed that their method for the convex hull works for more general polygons in 3-space. Djidjev and Lingas [5] showed their claim yields also a linear-time upper bound on computing the Voronoi diagram of the vertices of a monotone histogram (i. e. for sites sorted by their x-coordinates that have, in this order, monotone y-coordinates). On the other hand, the latter authors showed that it takes time $\Omega(n \log n)$ to compute the Voronoi diagram of the vertices of a non-monotone histogram (i. e. of a set of sites sorted by one coordinate.)

The aforementioned method of Aggarwal et al. is not complicated algorithmically. However it is quite non-trivial to follow and also it seems to involve large constants. On the other hand, already in 1986 Chew presented an extremely simple linear-time randomized algorithm for the Voronoi diagram of a convex polygon [3]. About 1990, Seidel showed a very simple complexity analysis of Chew's algorithm as an example of the so called backward analysis [14]. In 1991, Devillers presented an $O(n \log^* n)$-time randomized algorithm for the so called skeleton of a simple polygon, i.e. the Voronoi diagram of a simple polygon where the edges of the polygon are sites [4]. In 1992, Klein and Lingas presented a linear-time randomized algorithm for the so called bounded Voronoi diagram of a simple polygon [8]. Their algorithm combines Chew's idea with polygon decomposition and Voronoi diagram merging techniques. Another generalization has been mentioned by Rasch [13] who observed that even an abstract Voronoi diagram [10] can be constructed in linear time, by a randomized algorithm, if, for all subsets of point sites, the diagram has the shape of a tree and if the order of regions "at infinity" is given. In a forthcoming paper [11] the authors are presenting a deterministic linear algorithm for this task. Also, Yap has considered the generalization of the linear-time deterministic method due to Aggarwal *et al.* to inlude a convex set of points and circural segments [15].

In this note first we observe that Chew's algorithm can be easily generalized to include computing the convex hull of the polygons in 3-space that are subject of the method of Aggarwal *et al* and its generalization. In this way we obtain a simple linear-time randomized algorithm for the Voronoi diagram of any point sequence for which the method of Aggarwal *et al* and its generalization imply a linear-time upper bound (e.g. for monotone histograms). In the second part of the note, we directly generalize Chew's algorithm to include monotone sequence of straight-line segment sites (avoiding the reduction to the convex hull problem in 3-space). In result we obtain the first linear expected-time algorithm for the above problem.

2 The generalization of Chew's algorithm in 3-space

Guibas and Stolfi defined the so called *lifting mapping* μ of the Oxy plane into E^3 by $\mu(x,y) \rightarrow (x,y,x^2+y^2)$ [7].

Fact 2.1 (see [6]) *Let S be a finite set of points in the XY plane. The perpendicular projections of the edges of the lower part of the convex hull of $\mu(S)$ on the XY plane are the edges of the Delaunay triangulation of S. The analogous correspondance holds between the edges of the upper part of the convex hull of $\mu(S)$ and the edges of the so called furthest point-site triangulation. Given the convex hull of $\mu(S)$, we can compute the Delaunay triangulation of S and the furthest point Delaunay triangulation of S in linear time.*

Aggarwal *et al.* used the above fact to derive their linear upper bound on the time needed to construct the Voronoi diagram of the vertices of a convex polygon Q by proving that the convex hull of $\mu(Q)$ can be constructed in linear time [1]. In fact, they also claimed the following.

Claim 2.1 [1] *Let P be a polygon $(p_1,...,p_n)$ in E^3. Suppose that for each edge of any subpolygon P' of P given by a subsequence of $(p_1,...,p_n)$ there exists a plane that contains the edge and leaves all other vertices of P' in the same open half-space. The convex hull of the vertices of P can be constructed in time $O(n)$.*

Unfortunately, the proof of Fact 2.1 doesn't seem to be a straight-forward generalization of the convex case where the fact that the dual of the lower hull is a tree is used. Also, the algorithm due to Aggarwal *et al.* is rather conceptually involved. On the other hand, the convex hull of P can be constructed by the following, extremely simple algorithm. It can be seen as a natural generalization of Chew's randomized algorithm for the Delaunay triangulation of a convex polygon [3].

Algorithm 1

Input: A polygon P in space (with no four co-planar vertices) satisfying the assumptions from Claim 6.1.
Output: The convex hull $CH(P)$ of P.

1. If P has only three vertices then set $CH(P)$ to P;
2. If P has more than three vertices then pick a vertex q of P randomly, let p and r be its two neighboring vertices, and let P' be the polygon in space resulting from replacing the edges (p,q), (q,r) with (p,r);
3. Recursively compute the convex hull $CH(P')$ of P';
4. Transform $CH(P')$ to $CH(P)$

Lemma 2.1 *The transformation of $CH(P')$ into $CH(P)$ in Step 4 of Algorithm 1 can be done in time $O(a)$ where a is the number of edges of $CH(P)$ incident to q.*

Proof: By the assumptions on P, (p,r) is an edge of $CH(P')$ and (p,q) and (q,r) are edges of $CH(P)$. Also, q cannot lie inside $CH(P')$. Let D be the set of facets of $CH(P')$ that are not facets of $CH(P)$ (i.e. facets visible from q.) Observe that the straight-line dual to D is a connected graph (a tree). Also, at least one of the two facets adjacent to the edge (p,r) is in D. Therefore we can find D on $CH(P')$ as follows. First for each facet f adjacent to (p,r) in $CH(P')$ we test on which side of f the vertex v lies. In this way we can insert at least one facet in D. Then, for each facet f of $CH(P')$ that is adjacent to a facet already inserted into D and has not been yet tested we analogously determine the membership of f in D. After that we remove all the edges in $CH(P')$ between facets in D and for each vertex w of a facet in D we add the edge (q,w) to obtain $CH(P)$. As D can be regarded as a planar graph and each face in D is triangular by our assumptions on P, the total work done to construct $CH(P)$ is proportional to the number of vertices of D, which is $O(a)$. □

Theorem 2.2 *Algorithm 1 constructs the convex hull of P in linear expected time.*

Proof: Step 1 takes a constant time. To implement Step 2 in a constant time we keep a linear array A indexed by vertex numbers. An entry $A(i)$ is marked as passive if the vertex v_i doesn't occur in the current subpolygon P'. Otherwise two pointers to the two vertices adjacent to v_i in P' are kept in $A(i)$. Assume for a moment that no more than half of the elements of A are passive. To choose randomly q, a logarithmic number of random bits is used. If the corresponding entry is passive, a new logarithmic sequence of random bits is generated etc. By our temporary assumption, the expected number of trials in Step 2 until an appropriate q is found is $\leq \sum_{i=1}^{\infty} \frac{i}{2^i} = O(1)$. Thus, Step 2 can be implemented in expected constant time then. When half of the enties in A become passive, they are removed and the array is shrunk. The remaining "active" entries become reindexed appropriately. The above operation takes $O(n)$ time, and totally throughout the algorithm also $\sum_{i=1}^{\infty} \frac{i}{2^i} = O(n)$ time. Step 4 can be done in time $O(a)$ by Lemma 2.1. Since $CH(P)$ can be regarded as a planar graph, and q is randomly choosen, the expected value of a is $O(1)$. Since all steps in Algorithm 1 but for the recursive computation of $VB(P')$ take a constant expected time and P' has one less vertex than P, the whole algorithm runs in a linear expected time. □

In analogy to the corollaries from Claim 6.1 derived via Fact 6.1 in [1, 5], we obtain the following corollaries from Theorem 2.2.

Corollary 2.3 *There are simple algorithms with linear expected running-time for constructing:*
(i) the Delaunay triangulation or the Voronoi diagram of a convex polygon or a monotone histogram (with no four vertices co-cicular);
(ii) the intersection of the Voronoi diagram of a finite set of points (in general position), contained in the left half-plane and sorted by their Y coordinates, with the right half-plane.

Corollary 2.4 *The Voronoi diagram of a finite set of point sites (in general position) can be updated after deleting a single site in expected time proportional to the number of boundary edges of the region of the site.*

3 The direct generalization of Chew's algorithm in the plane

The conditions stated in Claim 6.1 assumed in Aggarwal et al's method and Algorithm 1 translate into the plane via Fact 2.1 as follows.

Lemma 3.1 *Let $p_0, p_2, ..., p_{n-1}$ be a sequence of points in the plane. The following conditions are equivalent:*
(i) The sequence under the μ mapping forms a polygon in space satisfying the requirements stated in Claim 2.1.
(ii) For any subsequence $p_{i_1}, p_{i_2}, ...,p_{i_k}$ of the sequence and $l = 1, ..., k-1$, $(p_{i_l}, p_{i_{l+1 \pmod k}})$ is an edge of either the Delaunay triangulation of the subsequence or the furthest-site Delaunay triangulation of the subsequence.

A generalization of the method of Aggarwal et al. or of Algorithm 1 and the above relationship to include more general form of sites, e.g. line segments, is technically not easy. For this reason it seems worthy to look for a direct generalization of Chew's algorithm on the plane that could include both more general sites and also the non-necessarily convex point configuration for which the convex hull methods work.
In fact, our direct algorithm works for line segment sites and could be easily extended to include more general form of sites. A point site is a special, degenerate case of a line segment site. The Voronoi diagram of a set of line segments is sometimes called *skeleton* [2]. The notion of Delaunay triangulation can be generalized to include line segments, for example, as follows.

Definition 2.1 *The Delaunay triangulation $Del(S)$ of a set S of disjoint line-segments is the straight-line dual to the Voronoi diagram of S where the vertex corresponding to the region of a line segment s is the middle point of s.*

Our randomized algorithm for the Voronoi diagram of a planar line segment set assumes the so called *search supporting function* implied by conditions similar to (ii).

Definition 2.2 *Let S be a finite set of line segments in the plane. A search supporting function for S is a function f from $2^S \times S$ to $(S \times S) \cup \{blank\}$ defined for any $U \subseteq S$ as follows:*

1. *for at least a constant fraction of $u \in U$ $f_1(U,u)$ is an edge of $Del(U - \{u\})$ such that at least a fragment of the corresponding edge of $Vor(U - \{u\})$ doesn't occur in $Vor(U)$;*
2. *for the remaining u in U, $f(U,u) = blank$.*

Algorithm 2

Input: A S of n disjoint line segments in the plane, a procedure for computing a search supporting function for S.
Output: $Vor(S)$ and $Del(S)$.

1. If S has no more than three line segments then compute $Vor(S)$ and $Del(S)$ directly;
2. If S has more than three line segments then pick a segment q in S randomly, and set S' to $S - \{q\}$.
3. Recursively compute $Vor(S')$ and $Del(S')$ keeping pointers between the corresponding edges of $Vor(S')$ and $Del(S')$;
4. Compute $f(U,u)$; if $f(U,u) = blank$ go to Step 2;
5. Transform $Vor(S')$ and $Del(S')$ to $Vor(S)$ and $Del(S)$ respectively.

Theorem 3.2 *Let S be a set of n disjoint line segments in the plane, and let f be a search supporting function for S. Suppose that for any $U \subseteq S$, and u in U, the time of computing $f(U,u)$ is not greater than $t(n)$. Algorithm 2 computes the Voronoi diagram of S in (expected) $O(t(n)n)$ time.*

Proof: Again, we should show that the transformation of Step 4 can be done in linear expected time. Let $f(U,u) = (v,w)$. By Step 3, we can compute the common boundary t of the regions of v and w in $Vor(U - \{u\})$ in constant time. By the properties of $f(U,u)$, t or at least a fragment of t will be covered by the region of u in $Vor(U)$. Compute $Vor(\{v,u,w\})$ (clearly in constant time). If the bisectors of v and u and the bisector of u and w in $Vor(\{v,u,w\})$ touch each other at a point of t then we can directly start building the boundary of the region of u in $Vor(U)$ (in other words merging $Vor(u)$ with $Vor(U - \{u\})$) following the two bisecors into opposite directions from the touching point in a standard way [9]. Otherwise the bisector of v and u intersect the boundary of the region of v in $Vor(U - \{u\})$ or the bisector of u and w intersects the boundary of the region of w in $Vor(U - \{u\})$ such that the set B of all boundary edges of the respective region between t and the intersection point forms a chain covered by the region of u in $Vor(U)$. We can find such an intersection in time $O(\#B)$ by alternating scan of the boundaries of the regions of u and w in $Vor(U - \{u\})$ starting from t in the two opposite directions on each boundary. In each phase of the scan we advance each of the four scanned chains by one boundary edge if such an intersection hasn't been found yet and the respective chain doesn't end with an infinite edge. Once such an intersection is found we can start building the boundary of the region of u in $Vor(U)$ in the standard way as in the first simpler case. The building takes time constantly bounded by the number $n(u)$ of the regions having a common boundary with the region of u in $Vor(U)$. Finding the intersection also takes time $O(n(u))$ since each edge in B corresponds to a different region having a common boundary with the region of u in $Vor(U)$ by the connectivity of the regions. While creating new region boundaries and removing some of the old ones during the transformation of $Vor(U')$ into $Vor(U)$ we correspondingly update $Del(U)$ in time $O(n(u))$. By the planarity of $Vor(U)$ and the random choice of u, $n(u)$ has expected value $O(1)$. Now to obtain the thesis it remains to observe that the expected number of trials until an u for which $f(U,u) \neq blank$ is $O(1)$ by our assumptions. □

By using the following lemma we can apply the above theorem to monotone line segment chains.

Lemma 3.3 *Let s_1, s_2, ...s_n be a sequence of disjoint line segments which induces a monotone polygonal chain. The partial function f defined for any subsequence $U = s_{i_1},...s_{i_k}$, and s_{i_l} where $l = 2,...,k-1$ by $f(U,u) = (s_{i_{l-1}}, s_{i_{l+1}})$ yields a search supporting function for the sequence.*

Proof: We may assume without loss of generality that the Y-coordinates of the line segments in the sequence monotonously grow with respect to the X-coordinate. Clearly the subsequence and the sub-subsequence have also the above property. Let $l = 2, ..., k-1$. Consider the middle m of the line segment s connecting the right endpoint of $s_{i_{l-1}}$ to the left endpoint of $s_{i_{l+1}}$. By the monotonicity of of the sub-subsequence $s_{i_1}, ...s_{i_{l-1}}$, $s_{i_{l+1}}, ...s_{i_k}$ and the disjointness of its elements the regions of $s_{i_{l-1}}$ and $s_{i_{l+1}}$ in the Voronoi diagram of the sub-subsequence have acommon boundary t passing through m. Thus, the edge $(s_{i_{l-1}}, s_{i_{l+1}})$ is in the Delaunay triangulation of the sub-subsequence. Again by the monotonicity and disjointness of the subsequence $s_{i_1}, ...s_{i_k}$, the segment s_{i_l} is closer to m than any other segment in the subsequence. Hence, at least a part of t has to disappear in the Voronoi diagram of the subsequence. Thus, the partial function specified in the thesis yields a search supporting function. □

Combing Theorem 3.2 with Lemma 3.3 we obtain our main application result.

Theorem 3.4 *Let S be a monotone finite sequence of disjoint line segments. The Voronoi diagram of S can be built in expected linear time.*

Proof: A search supporting function for the sequence is a trivial extension of the function $f(U, u)$ given in Lemma 3.3. The subsequence $U - \{u\}$ can be recursively chained during the performance of Algorithm 2. Therefore, the search supporting function can be computed in constant time. It remains to apply Theorem 3.2. □

References

[1] A. Aggarwal, L.J. Guibas, J. Saxe, and P.W. Shor. A Linear-Time Algorithm for Computing the Voronoi Diagram of a Convex Polygon. Discrete and Computational Geometry 2, 1987, Springer Verlag.

[2] F. Aurenhammer. Voronoi Diagrams—A Survey. Tech. Rep., Graz Technical University, 1988.

[3] P. Chew. Building Voronoi Diagrams for Convex Polygons in Linear Expected Time. Manuscript (1986).

[4] O. Devillers. Randomization yields simple $O(n \log * n)$ algorithms for difficult $\Omega(n)$ problems. International Journal of Computational Geometry and Applications, Vol2, No1 (1992), pp. 97-111

[5] H. Djidjev and A. Lingas. On Computing the Voronoi Diagram for Restricted Planar Figures. Proc. WADS'91, pp. 54-64, LNCS, Springer Verlag.

[6] H. Edelsbrunner. Algorithms in Combinatorial Geometry. EATCS Monographs on Theoretical Computer Science 10, 1987, Springer Verlag.

[7] L.J. Guibas and J. Stolfi. Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams. ACM Trans. Graphics 4, 1985, pp. 74–123.

[8] R. Klein and A. Lingas. A linear-time randomized algorithm for the bounded Voronoi diagram of a simple polygon. Proc. 9th ACM Symposium on Computational Geometry, San Diego, 1993.

[9] D.G. Kirkpatrick. Efficient computation of continous skeletons Proc. 20th IEEE Ann. Symp. on Foundations of Computer Science, 1979.

[10] R. Klein. Concrete and abstract Voronoi diagrams. LNCS 400, 1989.

[11] R. Klein and A. Lingas. Hamiltonian abstract Voronoi diagrams in linear time. Manuscript, 1993.

[12] F.P. Preparata and M.I. Shamos. Computational Geometry: An Introduction. Texts and Monographs in Theoretical Computer Science, Springer Verlag, New York, 1985.

[13] R. Rasch. Communicated by K. Mehlhorn, 1993.

[14] R. Seidel. Backwards Analysis of Randomized Geometric Algorithms. Manuscript, University of Berkeley, 1991.

[15] C. Yap and H. Alt Motion Planning in the CL-Environment.. Proc. WADS'89, Ottawa, Canada, LNCS 382, pp. 373-380.