An increasing-circle sweep-algorithm to construct the Delaunay diagram in the plane

B. Adam, P. Kauffmann, D. Schmitt and J.-C. Spehner *

Abstract

We present a new way to compute the Delaunay diagram of a planar set S of n sites in $O(n \ \log n)$ time by using a plane sweep technique. We sweep the plane by a circle whose center is a fixed point in the convex hull of S and whose radius increases from 0 to $+\infty$. This method is interesting notably when the diagram has to be constructed locally around a given point. We do not know of any method to reduce the sweep circle algorithm to a sweep line algorithm.
Keywords: Planar site Delaunay diagram, plane sweep algorithm.

1 Introduction

The Delaunay diagram, Delaunay triangulation and Voronoi diagram are well known structures in computational geometry and are used in various domains such as crystallography, physic, CAM-CAD, archaeology, ... [3, 5,...]. One of the major methods to construct these diagrams consists in sweeping the plane by a line. When the line sweeps over a new site, the current Voronoi diagram is updated by finding the region of this diagram in which the site is located [7]. Dually, it comes down to finding the site of the current Delaunay diagram to which the swept site must be connected [8].

In this paper, we introduce a new algorithm that computes the Delaunay diagram by sweeping the plane with an increasing circle. This algorithm updates the current Delaunay diagram by processing two kinds of events: the "site event" and the "ultimate point event".

We first define the location structures that will allow us to connect a newly swept site to the already validated Delaunay diagram. Then we show how the algorithm detects and processes the events.

Finally, we show that our algorithm constructs the Delaunay diagram in $O(n \log n)$ worst case running time and we compare it with the sweep line algorithm.

*Laboratoire MAGE, Université de Haute-Alsace, 68093 Mulhouse, France
{B.Adam|P.Kauffmann|D.Schmitt|spehner}@univ-mulhouse.fr

2 The definition of $Del(S)$

Let E be the Euclidean plane and $d(x, y)$ the Euclidean distance between two points x and y of E. Let S be a planar set of n points of E which are called *sites*.

Every circle σ of E that contains no site in its interior is said to be *S–critical*, and the set of sites which are on σ is called a *section*.

The Delaunay diagram of S is a partition of E whose *vertices are the sites of S*. For every section $\{s, t\}$, the open straight line segment $e(s, t)$ linking s to t is an *edge* of $Del(S)$. For every section Q of dimension 2, the interior $r(Q)$ of the convex hull of Q is a *region* of $Del(S)$. If $|Q| > 3$, $r(Q)$ is not a triangle and $Del(S)$ is not a triangulation.

The complementary of the convex hull of S is the only unbounded region of $Del(S)$.

3 The validated Delaunay diagram $Del(T, \rho)$ and its front $\mathcal{F}$

Let a point O in the convex hull of S be the origin of the polar coordinates and the center of the sweep circle C. S is sorted using the sites' radiuses. Two sites having the same radius are not ordered. If T is the set of swept sites, then $\forall t \in T, \forall s \in S \setminus T, \rho_t = d(O, t) < d(O, s) = \rho_s$ and the radius of the sweep circle $\mathcal{C}$ is $\rho = \max\{d(O, t); t \in T\}$. Every T–critical circle contained in $\mathcal{C}$ is also S–critical.

An edge $e(s, t)$ [resp. a region $r(Q)$] of $Del(T)$ is said to be *validated* if there exists a T–critical circle contained in $\mathcal{C}$ that goes through s and t [resp. all sites of Q]. The sub-diagram $Del(T, \rho)$ of $Del(T)$ built with the sites of T and the validated edges and regions of $Del(T)$ is called the *validated Delaunay diagram of* (T, ρ).

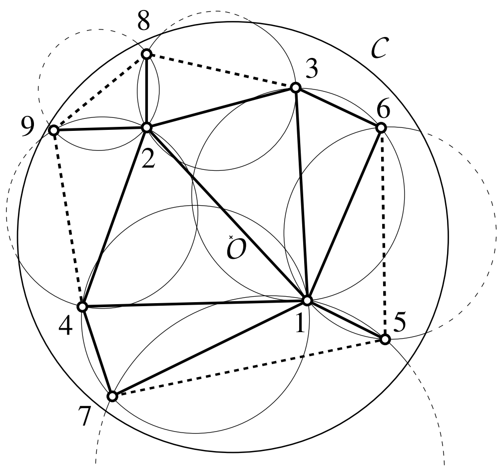

Figure 1: The validated Delaunay edges in full lines and the non–validated edges in dashed lines. The sites are numbered according to their distance to O. Front $\mathcal{F}$ is (1, 5, 1, 7, 4, 2, 9, 2, 8, 2, 3, 6, 1).

For every edge $e(s,t)$ of $Del(T,\rho)$, st denotes the oriented straight line segment of origin s and end point t. Every edge and every vertex of the unbounded region of $Del(T,\rho)$ is said to be *frontal*. If pq and qr are two consecutive edges of this unbounded region, the triple (p,q,r) is said to be *frontal*. The sequence $\mathcal{F}$ of all consecutive frontal edges is called the *Delaunay diagram front*.

4 The location of a point of $\mathcal{C}$ in $\mathcal{F}$

When $\mathcal{C}$ increases and sweeps over a site s, we search a site t in $\mathcal{F}$ such that s and t can be linked together to form a validated Delaunay edge. Therefore, we define a partition of $\mathcal{C}$ which is dual to $\mathcal{F}$ and we show how to locate a site s in this partition.

4.1 Definitions of the point and the arc attached to a frontal edge

If pq is a frontal edge, there exists a unique circle $C(p,q)$ going through p and q that is tangent to $\mathcal{C}$ at a point on the left of pq and such that $C(p,q)$ is contained in $\mathcal{C}$. This contact point is called the *point attached* to the edge pq and is denoted by $pat(p,q)$.

Lemma 1 *The circle $C(p,q)$ is S–critical. Hence, if $pat(p,q)$ is a site, $pqpat(p,q)$ is a validated Delaunay region.*

Let (p,q,r) be a frontal triple, the set of points of $\mathcal{C}$ that are on the left of $pat(p,q)pat(q,r)$ is called the *arc attached* to the triple (p,q,r) and is denoted by $arcat(p,q,r)$. $arcat(p,q,r)$ is an open circular arc.

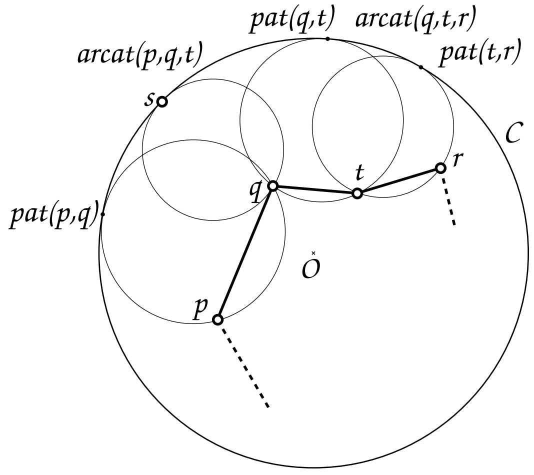

Figure 2: The frontal partition of $\mathcal{C}$ dual to $\mathcal{F}$.

Lemma 2 *For every point s of $arcat(p,q,r)$, the circle $D(q,s)$ tangent to $\mathcal{C}$ at s that goes through q is S–critical. Hence, if $s \in S$, the open straight line segment qs is a validated Delaunay edge.*

According to these two lemmas each newly swept site is linked to the validated Delaunay diagram. This proves that $Del(T,\rho)$ is connected.

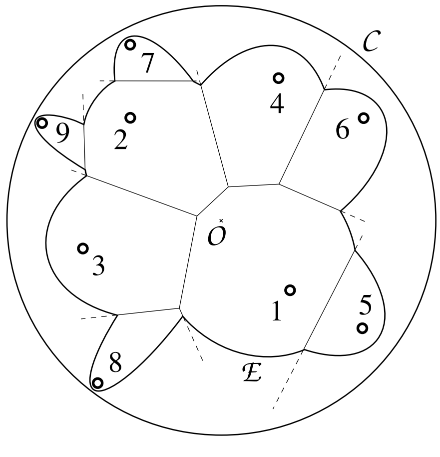

Figure 3: The elliptic front and the definitively constructed restriction of $Vor(S)$ (full lines).

Remark. For every $arcat(p,q,r)$, the set of the centers of the circles $D(q,s)$ is an elliptic arc. The restriction of $Vor(S)$ that is inside the union $\mathcal{E}$ of all the elliptic arcs is definitively constructed. $\mathcal{E}$ corresponds to the parabolic front of the sweep line algorithms [4, 8, 9].

4.2 The frontal partition of $\mathcal{C}$ and the location test

The partition of $\mathcal{C}$ in vertices that are the attached points, and in arcs that are the attached arcs, is called the *frontal partition of* $\mathcal{C}$. The mapping ψ from $\mathcal{F}$ to the frontal partition of $\mathcal{C}$ such that, for every frontal edge pq, $\psi(pq) = pat(p,q)$ and, for every frontal triple (p,q,r), $\psi(p,q,r) = arcat(p,q,r)$ is a one–to–one duality by lemma 1 and 2.

To locate a point s in the frontal partition of $\mathcal{C}$, we use a balanced binary search tree $BT(\mathcal{F})$. To each frontal edge is associated a node of $BT(\mathcal{F})$. A frontal edge is inserted in $BT(\mathcal{F})$ when created and is removed when it is no longer frontal. The frontal edges are sorted accordingly to their order in $\mathcal{F}$. Thus we can locate a point s of $\mathcal{C}$ in logarithmic time in the number of frontal edges. The location test is used in section **??** to create Delaunay edges.

5 Collision of attached points and determination of Delaunay regions

If (p,q,r) is a frontal triple such that r is on the left of pq, the circumcircle of the triangle pqr is denoted by $circ(p,q,r)$. The point of $circ(p,q,r)$ which is farthest from O is called the *ultimate point* associated to the frontal triple (p,q,r) and is denoted by $ult(p,q,r)$. $ult(p,q,r)$ is the last point of $circ(p,q,r)$ to be swept and stays *alive* until it is *killed* when (p,q,r) is no longer frontal. When the radius of $\mathcal{C}$ increases, $pat(p,q)$ and $pat(q,r)$ tend towards each other. If $circ(p,q,r)$ is S–critical, they collide in $ult(p,q,r)$; in this case $ult(p,q,r)$ is said to be validated and $circ(p,q,r)$ circumscribes a validated Delaunay region.

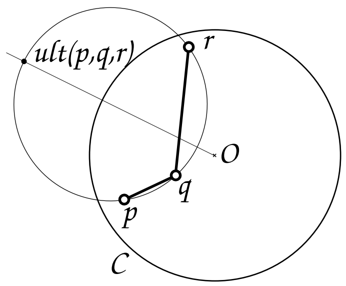

Figure 4: The ultimate point associated to (p,q,r).

If $k \geq 4$ consecutive frontal sites $s_i, s_{i+1}, \ldots, s_{i+k-1}$ are cocircular, $k-2$ triples of $\mathcal{F}$ have overlapping ultimate points. This simple test detects all cocircular sites and is used to build the non triangular regions.

To be able to find the ultimate point closest to O, the ultimate points are inserted in a balanced binary search tree $BT(\mathcal{U})$. An ultimate point is inserted in $BT(\mathcal{U})$ when created and is deleted from $BT(\mathcal{U})$ when killed or validated.

6 Processing the algorithm events

An event is a moment in the algorithm when $\mathcal{C}$ sweeps over a site or an ultimate point. The events add Delaunay edges and regions to $Del(T,\rho)$. Thus we need to update the trees $BT(\mathcal{F})$ and $BT(\mathcal{U})$.

6.1 The event "site which is not an ultimate point"

6.1.1 Updating $BT(\mathcal{F})$

If $s \in arcat(p,q,r)$, the Delaunay edge $e(s,q)$ is created in $BT(\mathcal{F})$. Otherwise s overlaps the point attached to a frontal edge pq. In this case the edge pq is unique and the Delaunay edges $e(s,p)$ and $e(s,q)$ are created in $BT(\mathcal{F})$. The former case implies the creation of a Delaunay region.

6.1.2 Updating $BT(\mathcal{U})$

If $s \in arcat(p,q,r)$, the ultimate point associated to the triple (p,q,r) is killed, if it exists. Otherwise s overlaps $pat(p,q)$. If p' and q' are respectively the sites preceding p and following q in $\mathcal{F}$, the ultimate points associated to the triples (p',p,q) and (p,q,q') are killed if they exist.

Moreover, if $s \in arcat(p,q,r)$ [resp. $s = pat(p,q)$], the new frontal triples (p,q,s) and (s,q,r) [resp. (p',p,s) and (s,q,q')] may generate new ultimate points.

6.2 The event "ultimate point which is not a site"

When sweeping an ultimate point u, there exists a sub–path $(p_1, p_2, \ldots, p_k)$ of $\mathcal{F}$ with $k \geq 3$ in which all the frontal triples have ultimate points overlapping u.

6.2.1 Updating $BT(\mathcal{F})$

We create the Delaunay edge $e(p_1, p_k)$ in $BT(\mathcal{F})$ which closes the Delaunay region of vertices $p_1, p_2, \ldots, p_k$. We delete from $BT(\mathcal{F})$ the edges $p_i p_{i+1}, \forall i \in [1, k-1]$.

6.2.2 Updating $BT(\mathcal{U})$

The ultimate points associated to the triples $(p_i, p_{i+1}, p_{i+2}), \forall i \in [1, k-2]$, are killed. The new frontal triples (p_0, p_1, p_k) and (p_1, p_k, p_{k+1}) may generate new ultimate points.

6.3 The event "ultimate site"

This event is a mix of the two previous ones and occurs when a site s overlaps ultimate points $ult(p_i, p_{i+1}, p_{i+2}), \forall i \in [1, k-2]$.

6.3.1 Updating $BT(\mathcal{F})$

We create the Delaunay edges $e(p_1, s)$ and $e(s, p_k)$ in $BT(\mathcal{F})$ that close the Delaunay region of vertices $s, p_1, p_2, \ldots, p_k$. Thus we delete from $BT(\mathcal{F})$ the edges $p_i p_{i+1}, \forall i \in [1, k-1]$, that are no longer frontal.

6.3.2 Updating $BT(\mathcal{U})$

We kill the ultimate points associated to $(p_i, p_{i+1}, p_{i+2}), \forall i \in [1, k-2]$. The new frontal triples (p_0, p_1, s) and (s, p_k, p_{k+1}) may generate new ultimate points.

6.4 The algorithm main lines

The sites are stored in a sorted list $Lst(\mathcal{S})$ following the increasing radius order. We use a map [6] to represent $Del(T, \rho)$. The algorithm ends when $\mathcal{C}$ has swept all the sites ($Lst(\mathcal{S}) = \emptyset$) and when all the Delaunay regions are constructed ($BT(\mathcal{U}) = \emptyset$).

Delaunay_diagram ($Lst(\mathcal{S})$:Sorted sites)$\rightarrow \mathcal{M} : Map$
  $BT(\mathcal{U}) \leftarrow \emptyset$; $BT(\mathcal{F}) \leftarrow \emptyset$; $M \leftarrow \emptyset$;
  while $Lst(\mathcal{S}) \cup BT(\mathcal{U}) \neq \emptyset$
    extract the minimum e of $Lst(\mathcal{S}) \cup BT(\mathcal{U})$;
    if e is a site which is not an ultimate point then
      locate e in $BT(\mathcal{F})$;
      if $e = pat(p, q)$ then
        create the edges ep and eq in $\mathcal{M}$;
      otherwise create the edge eq in $\mathcal{M}$;
    otherwise if e is an ultimate site then
      create the edges ep_1 and ep_k in $\mathcal{M}$;
    otherwise
      create the edge $p_1 p_k$ in $\mathcal{M}$;
    update $BT(\mathcal{F})$;
    update $BT(\mathcal{U})$;
  return $\mathcal{M}$;

7 The algorithm complexity

Lemma 3 *The number of frontal edges of F and the number of ultimate points alive are less than or equal to $2|T| - 2$.*

Proof. From 6, when a site is swept, except for the first one, at most two frontal edges are created. When an ultimate point which is not a site is swept, at least one frontal edge is removed. Thus the number of frontal edges is less than or equal to $2|T| - 2$. The number of ultimate points alive is also less than or equal to $2|T| - 2$ since every ultimate point is associated to a frontal triple.□

Theorem 1 *If S is a planar set of n sites, the increasing circle algorithm computes $Del(S)$ in $O(n \log n)$ time.*

Proof. Since $Del(S)$ is a partition of the plane with n vertices, $Del(S)$ admits at most $2n - 4$ regions. Using 6.1, 6.2 and 6.3, for every ultimate point swept, a face is validated and, for every site swept, a vertex is validated. Thus the algorithm handles at most $3n - 4$ events. Using lemma 3, every search, insertion or deletion in $BT(\mathcal{U})$ and in $BT(\mathcal{F})$ is done in $O(\log n)$ time. Therefore the construction of $Del(S)$ is done in $O(n \log n)$ time. Moreover, using the duality between $Vor(S)$ and $Del(S)$, $Vor(S)$ can be deduced in $O(n)$ time.□

8 Discussion

A question is: Is it possible to reduce our algorithm to a sweep line algorithm? This cannot be done with an inversion of pole I since it transforms parallel lines into circles tangent at I.

Let be the mapping

$$\phi : (\rho, \theta) \mapsto (x', y') = \left(-\rho, \frac{2\rho \sin\theta}{1 - \cos\theta}\right)$$

where (ρ, θ) are the polar coordinates of a point s of E and (x', y') the Cartesian coordinates of $\phi(s)$. ϕ transforms the circles centered at O in a set of parallel straight lines but does not transform $Del(S)$ in $Del(\phi(S))$.

We do not know of a better transformation that verifies this property.

Practical experiments show that the implementation of this algorithm is slower (~20%) than the sweep line algorithm. In our algorithm, the size of the front is bigger. This can be explained by the fact that the number of vertices of the convex hull of a set of n sites uniformly distributed in a circle is in $O(\sqrt[3]{n})$ but this number is in $O(\log n)$ for sites uniformly distributed in a rectangle.

However, the size of the binary tree of the ultimate points can be reduced by about a half. Only the locally minimal ultimate points, that is, those whose distance to O is less than that of their two neighbors in $\mathcal{F}$, are inserted in the tree.

The relative execution times of the different steps of the algorithm are:
– ultimate point and ultimate site events: 38 %
– site events: 47 %
– sorting and memory management: 11 %
– updating of the map: 4 %

9 Conclusion

We have given a new way to compute the Delaunay diagram of a set S of n sites using an increasing circle in $O(n \log n)$ time. This algorithm is slower than the sweep line algorithm but it is practical if we have to built the Delaunay diagram only locally around a given point.

The sweep circle method has also been used to construct convex hulls and farthest point Delaunay diagrams in the plane. In these cases a shrinking circle is used and the algorithms generally end without sweeping all the sites [1, 2].

References

[1]B. Adam, P. Kauffmann, D. Schmitt, J.-C. Spehner, *Sweep algorithms for planar convex hulls*, Technical Report, Laboratoire MAGE, Université de Mulhouse (1996).

[2]B. Adam, P. Kauffmann, D. Schmitt, J.-C. Spehner, *A shrinking circle algorithm for the planar farthest site Delaunay diagram*, Technical Report, Laboratoire MAGE, Université de Mulhouse (1996).

[3]F. Aurenhammer, *Voronoi diagrams – A survey of a fundamental geometric data structure*, A.C.M. Computational Surveys, 23, 3, (1992), p 345–405.

[4]D. Beauquier, J. Berstel et Ph. Chrétienne, *Éléments d'algorithmique*, Masson, (1992), p 411–425.

[5]J.-D. Boissonnat et M. Yvinec,*Géométrie Algorithmique*, Ediscience International (1995).

[6]J. Edmonds, *A combinatorial representation for polyhedral surfaces*, Notices Amer. Math. Soc. 7, (1960), 646.

[7]S. Fortune, *A sweepline algorithm for Voronoi diagrams*, Algorithmica, 2, (1987), p153–174.

[8]P. Kauffmann et J.-C. Spehner, *Sur l'algorithme de Fortune*, Revue internationnale de CFAO et d'informatique graphique, Volume 10 - n°4/1995, p 321–336.

[9]R. Seidel, *Constrained Delaunay triangulations and Voronoi diagrams with obstacles*, in Rep. 260 Graz, Austria, (1988), p178–191.