On Deletion in Delaunay Triangulation

Olivier Devillers

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

On Deletion in Delaunay Triangulation.

Olivier Devillers

N° 3451

Juillet 1998

THÈME 2

ISSN 0249-6399

INRIA
SOPHIA ANTIPOLIS

On Deletion in Delaunay Triangulation.

Olivier Devillers

Thème 2 — Génie logiciel
et calcul symbolique
Projet Prisme

Rapport de recherche n° 3451 — Juillet 1998 — 14 pages

Abstract: This paper present how space of spheres and shelling can be used to delete efficiently a point from d-dimensional triangulation. In 2-dimension, if k is the degree of the deleted vertex, the complexity is $O(k \log k)$, but we notice that this number apply only to low cost operations; time consuming computations are done only a linear number of times.

This algorithm can be viewed as a variation of Heller algorithm [Hel90, Mid93] which is popular in the geographic information system community. Unfortunalty Heller algorithm is false as explained in this paper.

Key-words: computational geometry, geometric computing, Delaunay triangulation, dynamic algorithms.

This work was partially supported by ESPRIT LTR 21957 (CGAL)

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, B.P. 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : 04 92 38 77 77 - International : +33 4 92 38 77 77 — Fax : 04 92 38 77 65 - International : +33 4 92 38 77 65

Suppressions dans la triangulation de Delaunay.

Résumé : Cet article montre comment l'espace des sphères et l'effeuillage (*shelling*) peuvent être utilisés pour implémenter efficacement la suppression d'un point dans la triangulation de Delaunay. En dimension 2, si k est le degré du sommet supprimé, la complexité est de $O(k \log k)$. On peut remarquer que cette complexité ne s'applique qu'à des opérations assez bon marché, les calculs les plus coûteux n'étant qu'en nombre linéaire.

Cet algorithme peut être vu comme une variation d'un algorithme proposé par Heller [Hel90, Mid93] populaire dans le domaine des systèmes d'information géographique. Malheureusement l'algorithme original de Heller est faux.

Mots-clés : géométrie algorithmique, calcul géométrique, triangulation de Delaunay, algorithmes dynamiques.

1 Introduction

The computation of the Delaunay triangulation of a set S of n points in the plane is one of the classical problems in computational geometry.

Many structures and algorithms have been proposed in the past to compute Delaunay triangulations. Some of these algorithms have the two following properties: they are incremental and they do not used complicated data structure in addition to the triangulation itself. Among these algorithms we can cite the historical algorithm of Green and Sibson [GS78], or some other variants [MSZ96, BD95, Dev98, DLM98, Lem97]. All walk in the triangulation to accelerate point location.

The advantage of that category of Delaunay incremental algorithm is that they can be easily turned in fully dynamic Delaunay algorithm. Since there is no complicated data structure for point location, the deletion of a point is reduced to the deletion in the triangulation itself.

Definition and notations

Given a set $\mathcal{S}$ of points in d dimensional space, $\mathcal{DT}(\mathcal{S})$ the Delaunay triangulation of $\mathcal{S}$ is defined by the following property: *$d+1$ points of $\mathcal{S}$ are the vertices of a Delaunay simplex if and only if the sphere passing through these points does not contain any point of $\mathcal{S}$* (see Figure 1 for a Delaunay triangulation in two dimensions).

Given the Delaunay triangulation $\mathcal{DT}(\mathcal{S})$ and a vertex p in $\mathcal{DT}(\mathcal{S})$, we address the problem of finding $\mathcal{DT}(\mathcal{S} \setminus \{p\})$.

In two dimension, the natural parameter to evaluate the complexity of this problem is the degree k of p in $\mathcal{DT}(\mathcal{S})$ since the deletion of p means that k triangles must be removed from the triangulation and $k-2$ new triangles must be created to fill this hole. In the worst case, k can be linear, but if p is chosen randomly in $\mathcal{S}$, then it is well known that the expected value of k is 6, without any assumptions on the point distribution.

In higher dimension, the number of simplices incident to p is not directly related to the number of created simplices to fill the hole; we will let f denote the sum of these two numbers. In the worst case, the whole Delaunay triangulation can be affected and $f = O(n^{\lfloor \frac{d+1}{2} \rfloor})$. This distribution does not reflect practical configurations and a constant value of f is more likely in practice. For a uniform point distribution, the expected value of f can be proven to be constant. In three dimension, the expected number of deleted tetrahedron for Poisson distribution is $\frac{96}{35}\pi^2 \simeq 27$ [OBS92].

Thus, even if we do not want to neglect the possible case of big value of k, we have to keep in mind that a good algorithm must perform well on small value of k.

Previous related work

Classical computational geometry have already addressed the problem of deleting point in Delaunay triangulation. This can be done with optimal asymptotic complexity $O(k)$ in 2 dimensions [Che87, AGSS89]. But these algorithms are a little bit too intricate and the big O hide a too important constant to be good algorithms for reasonable values of k (see Section 5.3).

Practitioners often prefer algorithmic simplicity to theoretical optimality, and prefer simple suboptimal $O(k^2)$ implementation of the deletion algorithm. It can be done, for example by flipping to reduce the degree of the deleted vertex to 3, and flipping again to restore the Delaunay property. Another simple algorithm consists in finding the Delaunay triangle incident to an edge of the hole in $O(k)$ time which also yields an $O(k^2)$ time algorithm.

An very simple $O(k \log k)$ solution have been suggested by Heller [Hel90, Mid93], where ears of the hole are filled in turn. In that way during the algorithm we always have a simple polygon of decreasing size to triangulate. Unfortunately, this solution is wrong, but we will show in this paper how to correct it.

Overview

In this paper, we provide a very simple and efficient $O(k \log k)$ algorithm to delete a vertex in a planar Delaunay triangulation based on shelling [BM71, Sei86] and duality [DMT92, Ped70]. We also discuss the effective complexity of this algorithm and of a few others for small values of k. We will study the different kinds of geometric predicates necessary for these algorithms.

This algorithm well generalize in higher dimensions, its time complexity becomes $O(f \log f)$ where f is the number of created tetrahedron, and it generalizes also to regular triangulations (power diagrams).

2 Two dimensional algorithm

A deletion algorithm has to remove all triangles incident to p and retriangulate the star-shaped polygon $H = \{q_0, q_1, \ldots, q_{k-1}, q_k = q_0\}$ created by these removals (see Figure 1).

2.1 Ears, and a wrong algorithm

We first define what an ear of a polygon is. Three consecutive vertices along H boundary $q_i q_{i+1} q_{i+2}$ are said to form an ear of H if the line segment $q_i q_{i+2}$ is inside H and does not intersect its boundary. An ear of H will be said Delaunay, if the circle passing through $q_i q_{i+1} q_{i+2}$ does not contain any other vertices of H. Heller

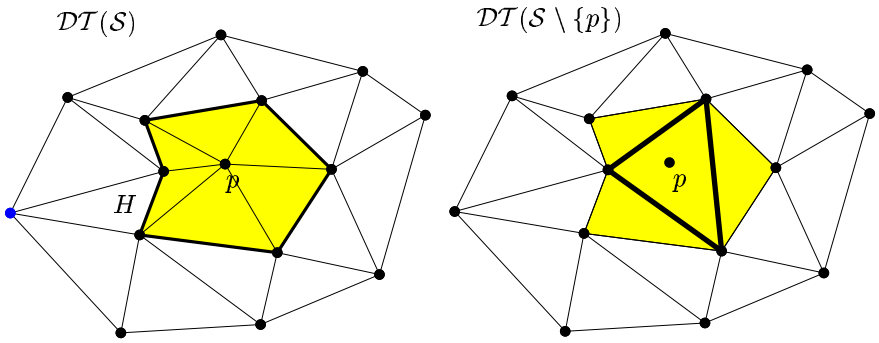

Figure 1: Deletion of a vertex.

[Hel90] and Midtbø [Mid93] claimed (without proof) that among all the potential ears $q_i q_{i+1} q_{i+2}$ of H the ear having the circumcircle of smallest radius is a Delaunay ear. This claim is false as illustrated by Figure 2, the left of Figure 2 shows the Delaunay triangulation $\mathcal{DT}(\mathcal{S})$ and the hole H to be retriangulated; the right of the figure shows two potential ears $q_0 q_1 q_2$ and $q_1 q_2 q_3$. Ear $q_0 q_1 q_2$ is the one with smallest circumcircle among all ears of H, but it contains q_3 which invalidates Heller and Midtbø claim. The error of Heller and Midtbø is to assume that when we deform a circle through $q_1 q_2$ to grow its part inside H the radius is increasing, but this is true only if the center of the circle is inside H which is not the case in Figure 2.

In fact, the idea of finding an ear that belongs to the Delaunay triangulation works, but with another criterion as explained below.

2.2 Delaunay and convex hull

There exist a well known duality between Delaunay triangulation in dimension d and convex hull in dimension $d+1$ [Aur87, DMT92]. If we associate to a point $p = (x, y) \in \mathcal{S}$ a point $p^\star = (x, y, x^2+y^2)$ on the paraboloid Π of equation $z = x^2+y^2$, the Delaunay triangulation of $\mathcal{S}$ is the projection of the convex hull of the 3D points. The reason is that for $p, q, r, s \in \mathcal{S}$, p is inside the circle C_{qrs} through qrs if and only if $p^\star$ is below the plane P_{qrs} through $q^\star r^\star s^\star$ ($\Pi \cap P_{qrs}$ projects on circle C_{qrs}). We even have the equality between the power of p with respect to C_{qrs} and the signed vertical distance between $p^\star$ and P_{qrs}.[1]

[1] If C is a circle of center x and radius r, p is a point and l is a line through p intersecting C in t and u, then $power(p, C) = |xp|^2 - r^2 = \overline{pt}\,\overline{pu}$ where $\overline{yz}$ is the signed length of yz. $power(p, C)$

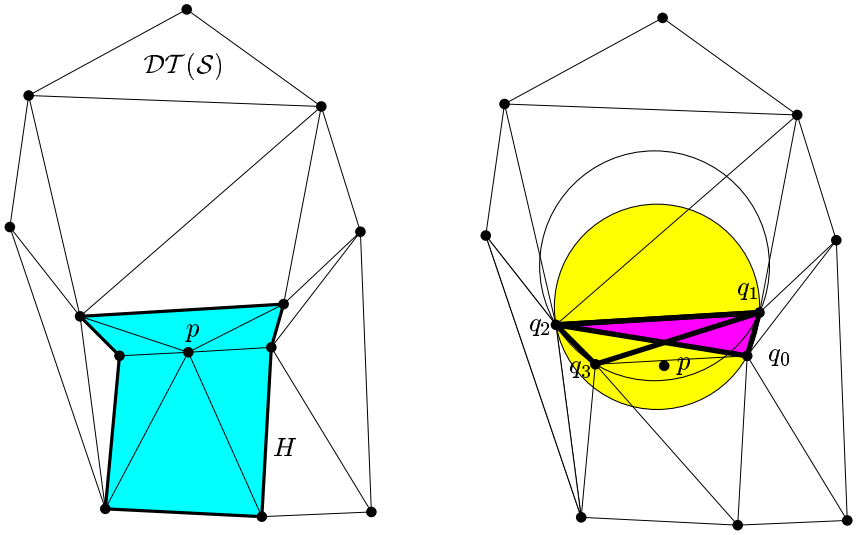

Figure 2: The smallest potential ear may not belongs to Delaunay. The shaded triangle is the ear with smallest radius, but it contains q and thus do not belongs to Delaunay.

2.3 Shelling

Convex hull can be computed by the shelling algorithm [Sei86]. The shelling [BM71] of a convex polyhedron P is the enumeration of its faces in some sympathetic order. We imagine that an observer is moving along a line l going through the polyhedron, starting at the intersection of P and l. At the starting position, the observer can see only one face of P (the face intersecting its trajectory) and when it moves away from the P it discovers other faces one by one; at the infinity the observer see "half" of P. Then the observer restart at infinity on the opposite side of l (where it see the other half of P), and then moves on l enumerating the faces of P when they disappear from its view. This order is called the shelling order of P with respect to l; it has the property that the set of enumerated faces remain simply connected during the enumeration. Seidel algorithm for convex hull reports the faces of the convex hull in that order maintaining a priority queue of potential new faces. These new faces have two kinds depending if the next face contains a new vertex or links already visible vertices.

2.4 Deletion in Delaunay

We can now, reuse the idea of finding Delaunay ears to retriangulate the hole created by the deletion of a point p in $\mathcal{DT}(\mathcal{S})$. Using duality with convex hull, the problem is transformed in filling the hole in the convex hull created by the deletion of $p^\star$. Then we can remark that if we use the shelling order with respect to the vertical line through $p^\star$, an observer (going up) reaching $p^\star$ see exactly the boundary of this hole; thus the end of the shelling correspond exactly to the triangulation of the hole. This partial shelling is easier to implement than Seidel algorithm, since all the vertices are already visible and thus only one kind of potential new faces have to be found. The already noticed correspondence between vertical distance and power yields to the following lemma.

> **Lemma:** *Given a polygon* $H = \{q_0, q_1, \ldots, q_{k-1}, q_k = q_0\}$ *and a point* p *such that the edges of* q_iq_{i+1} *belongs to the Delaunay triangulation of* $\{q_0, q_1, \ldots, q_{k-1}, p\}$. *If* $|power(p, circle(q_i, q_{i+1}, q_{i+2}))|$ *is minimal, then* q_iq_{i+2} *is an edge of the Delaunay triangulation of* $q_0, q_1, \ldots, q_{k-1}\}$

is zero on C boundary, negative inside and positive outside. When C is given by three points, the power can be computed without computing the radius as explained in Section 3.1. Power is negative inside the circle and positive outside.

Thus the deletion of a point can be implemented in a simple way, maintaining a structure to store the ears. The ears are naturally ordered along the boundary of the hole, and each have a priority. This structure must support the following operations : find next and previous ear, delete ear with minimum priority and modify the priority of an ear. This structure can be implemented with any dictionary structure augmented by next and previous pointers.

Algorithm *Delete*($\mathcal{DT}(\mathcal{S}), p$))

1. Let $q_0 q_1 \dots q_{k-1}$ the vertices incident to p in $\mathcal{DT}(\mathcal{S})$ in ccw order around p;
2. Let Q be a priority queue;
3. **for** $i = 0$ **to** $k - 1$
4. **do** $ear \leftarrow q_i q_{i+1} q_{i+2}$;
5. **if** counterclockwise($q_i q_{i+1} q_{i+2}$)
6. **then** $p \leftarrow \infty$; *//not an ear*
7. **else** $p \leftarrow -power(p, ear)$; *//inside circle power< 0*
8. Q.insert(p,ear); *//priority/key*
9. **while** size(Q) > 3
10. **do** $ear \leftarrow Q$.minimum();
11. create triangle ear and linked it to its two existing neighbors;
12. $ear0 \leftarrow ear$.previous;
13. $ear1 \leftarrow ear$.next;
14. $ear0$.vertex(2) $\leftarrow ear$.vertex(2); $ear0$.next $\leftarrow ear1$;
15. $ear1$.vertex(0) $\leftarrow ear$.vertex(0); $ear1$.previous $\leftarrow ear2$;
16. Q.delete(ear);
17. Q.modify-priority($ear0$);
18. Q.modify-priority($ear1$);
19. $ear \leftarrow Q$.minimum(); *//the three last ears are identical*
20. create triangle ear and linked it to its three existing neighbors;

Higher dimension The generalization to d dimensions is easy, the boundary of the region to retriangulate is a simple polyhedron H, and the ears are simplices formed by the vertices of two incident facets of H. The difference is that the same simplex may correspond to $O(d^2)$ pair of incident facets, and that the creation of an ear may modify $O(d^2)$ other ears in the priority queue.

3 2D Analysis

3.1 Power computation

An analytical expression of the power of p with respect to $q_0q_1q_2$ is

$$power(p, circle(q_0, q_1, q_2)) = \frac{\begin{vmatrix} x_{q_0} & x_{q_1} & x_{q_2} & x_p \\ y_{q_0} & y_{q_1} & y_{q_2} & y_p \\ x_{q_0}^2 + y_{q_0}^2 & x_{q_1}^2 + y_{q_1}^2 & x_{q_2}^2 + y_{q_2}^2 & x_p^2 + y_p^2 \\ 1 & 1 & 1 & 1 \end{vmatrix}}{\begin{vmatrix} x_{q_0} & x_{q_1} & x_{q_2} \\ y_{q_0} & y_{q_1} & y_{q_2} \\ 1 & 1 & 1 \end{vmatrix}}$$

The 3×3 determinant is the orientation test of $q_0q_1q_2$ and the 4×4 determinant the incircle test of p with respect to $q_0q_1q_2$. We have to notice first that if $q_0q_1q_2$ has the wrong orientation, then it is not an ear and thus the 4×4 determinant does not need to be computed, second that the orientation test is a minor of the incircle test and thus the power computation just needs one division in addition to the usual incircle test.

Using dynamic programming development of the determinant, the power computation need 14 additions, 15 multiplications and one division.

Notice if just q_2 changes, we do not need to recompute everything. In fact we can do it with 6 additions, 7 multiplications and one division.

3.2 Complexity

The theoretical asymptotic complexity, this algorithm is clearly $O(k \log k)$, but this complexity is involved only for the management of the priority queue which involve relatively cheap operations (pointer manipulations and comparisons of computed power).

Since the most expensive geometric operations are the power computations, we will count exactly the number of such operations. The initial size of the priority queue is k, and thus its initialization needs at most k power computation. Each ear creation, modified two other ears and thus two powers need to be recomputed, and the deletion is completed when the size of the queue is 3, thus the total number of power computations is $k + 2(k - 4) = 3k - 8$.

It is possible, as noticed above, to update the power of p with respect to $q_iq_{i+1}q_{i+2}$ when ear $q_{i+1}q_{i+2}q_{i+3}$ is processed. In the new polygon $H \setminus \{q_i + 2\}$, $q_iq_{i+1}q_{i+3}$ is an ear and the power of p with respect to $q_iq_{i+1}q_{i+3}$ can be obtained by updating

the power of p with respect to $q_i q_{i+1} q_{i+2}$for a cheaper cost. Then the total number of computations becomes $2k-4$ power computations and $k-4$ power updates.

3.3 Robustness issue and degeneracies

The algorithm presented above do not address robustness issues. If the floating point arithmetic is used to perform power computations, the results are rounded and their comparisons could be evaluated in a wrong way. We can first observe that the deletion algorithm will terminate even with incorrect arithmetic: it fill ears in turn and thus construct a topological triangulation, which can be non Delaunay, or even have a non-planar embedding. But, even if producing a non-exact Delaunay triangulation may be acceptable for the deletion algorithm, it is unacceptable for many insertion algorithms which are not able to process unexact triangulation.

The usual way to solve robustness issues consists in using exact arithmetic. To ensure good performance, we can use arithmetic filter to use exact computations in power comparisons only in difficult cases where the two powers are close. Some exact computations are used to take the right decision. This approach of filtering out easy cases has been proven efficient on the orientation and incircle tests [DP98]. Degeneracies can be solved using perturbation techniques [Sei98, ADS97].

4 Experimental results

4.1 Code and data

We implement this algorithm inside the simple hierarchical structure of the author [Dev98]. Various kind of data points have been tested. The Delaunay triangulation of the points is computed first, and then points are all removed in a random order. The coordinates are 24 bits integers and we use static and semi-static filters to ensure exactness of the evaluation of geometric predicates.

We used sets of size 1,000,000 points described in Figure 3.

- *random*: points evenly distributed in a square.
- *ellipse2*: 95% points evenly distributed on an ellipse plus 5% points evenly distributed in a square.
- *circle*: points evenly distributed on a circle.
- *parabola*: points evenly distributed on a parabola,

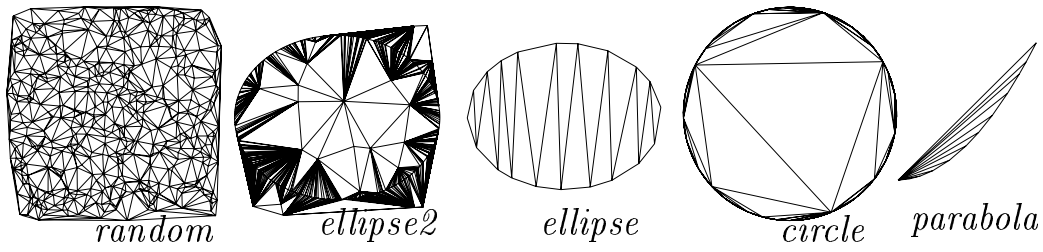

Figure 3: Data sets.

distribution	size	insertion time	deletion time
random	1,000,000	63s	36s
ellipse2	1,000,000	102s	136s
ellipse	1,000,000	123s	56s
circle	1,000,000	78s	56s
parabola	1,000,000	100s	87s

Figure 4: Running times

If the *circle* and *parabola* examples can be considered as pathological inputs, the *ellipse2* example is more realistic, Delaunay triangulation of points distributed on a curve occurs in practical applications, for example in shape reconstruction (see Figure 3).

4.2 Running times

Running times are given in Figure 4. They are obtained on a Sun Ultra1 200MHz 128Mo main memory. The code is written in C++ and compiled with AT-T compiler with optimizing options. Times have been obtained with the `clock` command and is given in seconds. The time that is measured is just for Delaunay computation; it does not take into account the time for input or output.

The more expensive running time for deletions on `ellipse2` data sets is mainly due to the fact that some points have an important degree (points which are random in the square but close to the ellipse).

5 Practical remarks and alternative methods

In two dimensions, some alternative methods are of some interest

5.1 Diagonal flipping

One of the interest of a method using diagonal flipping is that it does not introduce new geometric predicates, it uses only incircle tests and thus has lower degree and generalize easily to various metrics. However such a method may need $O(k^2)$ incircle tests in the worst case and it is not so simple to code efficiently. A good implementation of a flipping method [Sno98] will retriangulate H by basically linking all q_i to q_0 and flipping the edges turning around q_0. In the worst case, such a method may use $(k-3)+(k-4)+\ldots+1=\frac{(k-2)(k-3)}{2}$ incircle tests.

This flipping method is not so easy to code, an simpler solution consist in maintaining a queue of edges to be tested for potential flip. Each time the diagonal of a quadrilateral is flipped, the four edges of the quadrilateral are inserted in the queue. This simpler algorithm will make much more incircle tests since the flip are not performed in some relevant order as above.

5.2 Edge completion

A second method consists in finding q_i such that $q_0q_1q_i$ is a Delaunay triangle, which can be done in $k-3$ incircle tests, and triangulating recursively the two holes. In the worst case, the number of incircle tests is exactly the same that in the flipping method. An implementation of this technique is intrinsically recursive and thus imply some costs here.

5.3 Randomized algorithm

In Chew [Che87] randomized algorithm, each point to reinsert need an expected number of about 5 incircle tests, which yields a total of $5k+O(1)$.

5.4 On alternative methods

For small value of $k<9$, flipping or edge completion need less incircle tests computations, but the simplicity of our algorithm and its good performance for $k\geq 9$ make it a very good candidate for edge deletion. Nevertheless, it can be interesting to treat as special cases some small k values such as $k=4$ and $k=5$.

5.5 Higher dimensions

In higher dimensions, flipping, edge completion and shelling algorithms generalize but things became more difficult. The flipping must be done in higher dimension which made it more intricate to implement [ES96]. Edge completion transforms in facet completion and must deal with the triangulation of non simply connected polyhedron. Shelling is the most easy to generalize. Furthermore the increase in the average value of k with the dimension reinforce its advantage on alternative candidates in higher dimensions.

Code

A compiled demo version is available at http://www.inria.fr/prisme/logiciels/del-hierarchy/.

Acknowledgement

The author would like to thank Jack Snoeyink and Mariette Yvinec for helpful discussions and careful reading of this paper.

References

[ADS97] P. Alliez, O. Devillers, and J. Snoeyink. Removing degeneracies by perturbing the problem or the world. Research Report 3316, INRIA, 1997.

[AGSS89] A. Aggarwal, Leonidas J. Guibas, J. Saxe, and P. W. Shor. A linear-time algorithm for computing the Voronoi diagram of a convex polygon. *Discrete Comput. Geom.*, 4(6):591–604, 1989.

[Aur87] F. Aurenhammer. Power diagrams: properties, algorithms and applications. *SIAM J. Comput.*, 16:78–96, 1987.

[BD95] P. Bose and L. Devroye. Intersections with random geometric objects. Technical report, School of Computer Science, McGill University, 1995. Manuscript.

[BM71] H. Bruggesser and P. Mani. Shellable decompositions of cells and spheres. *Math. Scand.*, 29:197–205, 1971.

[Che87] L. P. Chew. Constrained Delaunay triangulations. In *Proc. 3rd Annu. ACM Sympos. Comput. Geom.*, pages 215–222, 1987.

[Dev98] O. Devillers. Improved incremental randomized Delaunay triangulation. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pages 106–115, 1998.

[DLM98] L. Devroye, C. Lemaire, and J.-M. Moreau. Binsearch and walk: a fast point location procedure in 2- and 3-dimensional on-line Delaunay triangulations, 1998. in preparation.

[DMT92] O. Devillers, S. Meiser, and M. Teillaud. The space of spheres, a geometric tool to unify duality results on Voronoi diagrams. In *Proc. 4th Canad. Conf. Comput. Geom.*, pages 263–268, 1992.

[DP98] O. Devillers and F. Preparata. A probabilistic analysis of the power of arithmetic filters. *Discrete and Computational Geometry*, 1998. à paraître.

[ES96] H. Edelsbrunner and N. R. Shah. Incremental topological flipping works for regular triangulations. *Algorithmica*, 15:223–241, 1996.

[GS78] P. J. Green and R. R. Sibson. Computing Dirichlet tessellations in the plane. *Comput. J.*, 21:168–173, 1978.

[Hel90] M. Heller. Trinagulation algorithms for adaptive terrain modeling. In *Proc. 4th Internat. Sympos. on Spatial Data Handling*, pages 163–174, 1990.

[Lem97] C. Lemaire. *Triangulation de Delaunay et arbres multidimensionnels.* thèse de doctorat en sciences, École des Mines de St-Etienne, France, 1997.

[Mid93] T. Midtbø. *Spatial Modelling by Delaunay Networks of Two and Three Dimensions.* Ph.D. thesis, Norvegian Institute of Technology, Trondheim, 1993.

[MSZ96] Ernst P. Mücke, Isaac Saias, and Binhai Zhu. Fast randomized point location without preprocessing in two- and three-dimensional Delaunay triangulations. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, pages 274–283, 1996.

[OBS92] Atsuyuki Okabe, Barry Boots, and Kokichi Sugihara. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams.* John Wiley & Sons, Chichester, UK, 1992.

[Ped70] D. Pedoe. *Geometry, a comprehensive course.* Dover Publications, New York, 1970.

[Sei86] R. Seidel. Constructing higher-dimensional convex hulls at logarithmic cost per face. In *Proc. 18th Annu. ACM Sympos. Theory Comput.*, pages 404–413, 1986.

[Sei98] R. Seidel. The nature and meaning of perturbations in geometric computing. *Discrete Comput. Geom.*, 19:1–17, 1998.

[Sno98] J. Snoeyink. Non redundant flip for point deletion in delaunay triangulation, 1998. Personnal communication.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - B.P. 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Lorraine : Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 Villers lès Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot St Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, B.P. 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399