Delaunay Triangulation of Imprecise Points, Preprocess and Actually Get a Fast Query Time

Olivier Devillers

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Delaunay Triangulation of Imprecise Points, Preprocess and Actually Get a Fast Query Time

Olivier Devillers

N° 7299

Mai 2010

ISSN 0249-6399 ISRN INRIA/RR--7299--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SOPHIA ANTIPOLIS - MÉDITERRANÉE

Delaunay Triangulation of Imprecise Points, Preprocess and Actually Get a Fast Query Time

Olivier Devillers

Thème : Algorithmique, calcul certifié et cryptographie
Équipe-Projet Geometrica

Rapport de recherche n° 7299 — Mai 2010 — 10 pages

Abstract: We propose a new algorithm that preprocess a set of n disjoint unit disks to be able to compute the Delaunay triangulation in $O(n)$ expected time. Conversely to previous similar results, our algorithm is actually faster than a direct computation in $O(n \log n)$ time.

Key-words: Delaunay triangulation, uncertainties, randomization

This work is partly supported by ANR grant Triangles (ANR-07-BLAN-0319).

Centre de recherche INRIA Sophia Antipolis – Méditerranée
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Triangulation de Delaunay de points dans des disques. Un algorithme vraiment rapide.

Résumé : Nous proposons un algorithme qui prétraite un ensemble de disques unitaires disjoints pour être capable de calculer la triangulation d'un ensemble de n points, un dans chaque disque, en temps moyen $O(n)$. Par rapport à d'autres résultats simiaires, notre algorithme permet également d'avoir effectivement un temps de calcul meilleur que les algorithmes classiques en $O(n \log n)$.

Mots-clés : Triangulation de Delaunay, incertitudes, randomisation

1 Introduction

A popular way to model geometric uncertainties in computational geometry, is to replace a data point by a region, e.g. a disk, where the point can be. We call such a region an *imprecise point*. An early work in that direction is the ϵ-geometry introduced by Guibas et al. [12]. Given a set of imprecise points, we call an *instance* a set of points, each one taken in the region of an imprecise point. Then, given a classical problem for a set of points, say the Delaunay triangulation computation, imprecise computational geometry may ask different questions:
-Strong problem- what are the pair of points that define a Delaunay edge for all possible instances ? (strongly Delaunay edge);
-Weak problem- what are the pair of points that define a Delaunay edge for at least one instance ? (weakly Delaunay edge); or
-Instance problem- can we preprocess the imprecise points and construct a data structure to be able to solve the problem on a particular instance in a fast way?

The strong and weak approaches have been used, e.g. for convex hull computations [13, 15], but a drawback of this approach is that it often involves intricate predicates. For example, if the original problem needs a simple predicate such as an orientation test on three points, the imprecise approach will test the position of a disk with respect to the tangent to two other disks.

In this paper, we will address the problem of answering instance queries for the Delaunay triangulation problem, where the imprecise points are disks in the plane.

Previous work

Imprecise points Löffler and Snoeyink [14] proposed an algorithm that preprocesses a set of disjoint unit disks in $O(n \log n)$ time and computes the planar Delaunay triangulation of an instance in $O(n)$ time. This algorithm has a reasonably simple description but uses as a building block the linear time construction of the constrained Delaunay triangulation of a simple polygon [5] which makes this result mainly theoretical. Then, Buchin et al. [3] proposed a simpler solution which uses the split of a Delaunay triangulation in linear time [4], this second solution remains a bit heavy in practice; indeed, the preprocessing compute a Delaunay triangulation of $8n$ points (center and boundary of the disks), then the points of the instance can be added in linear time to get a triangulation of $9n$ points and split it in the triangulation of the instance and the $8n$ points triangulation again. In the same paper, they also proposed a different algorithm based on quadtrees allowing overlapping disks of different radii.

Delaunay construction and walking strategies Let us recall here a useful classical incremental algorithm to compute the Delaunay triangulation. When a new point is inserted in a triangulation, the triangle containing it is located by "walking in the triangulation" from some starting point [9], then the triangulation is modified to remove non Delaunay triangles and fill the resulting hole with new triangles incident to the new point.

If the points are inserted in a random order, standard backward analysis [16] remarks that the last point is a random point, thus its expected degree is

less than 6 and the expected cost of modifying the triangulation is constant. Thus the remaining cost is the one of the walk which is directly related to the strategy to choose the starting point and there is a huge literature on randomized incremental construction to choose good starting points [8, 10, 11, 6, 1, 7, 2].

Contribution

In this paper we do not improve the theoretical asymptotic complexity, but we propose a so simple algorithm that its description fits in a dozen of lines. The algorithm works for any set of balls (overlapping, different radii, any dimension) but to yield to a good complexity, the analysis requires that the imprecise points are unit disks in the plane (possibly overlapping a constant number of times). This analysis can be extended in higher dimensions under some suitable hypotheses. For disks of different radii overlapping at most twice, we provide a pathological example where our algorithm reaches a quadratic behavior. For disjoint disks of different radii the analysis remains open.

Moreover, the algorithm is also efficient in practice and do not need any new predicates different that usual predicates for Delaunay triangulation. Our benchmarks conclude that we can answer a query much faster than with the Delaunay hierarchy [8, 17] and in a time better than spatial sorting [7, 2].

2 Notations

— For P a point set, let DT_P denote the Delaunay triangulation of P, NN_P its nearest neighbor graph and $NN_P(v)$ the nearest neighbor in P of $v \in P$.
— For a graph G, and a vertex v of G, $d^\circ_G(v)$ is the degree of v in G.
— If p denote an imprecise point represented by a disk, $\dot{p}$ denote the center of p and $\hat{p}$ an instance of p. Given a set of disks $S = \{p_1, p_2 \dots p_n\}$ (imprecise points), $\dot{S} = \{\dot{p}_1, \dot{p}_2 \dots \dot{p}_n\}$ and $\hat{S} = \{\hat{p}_1, \hat{p}_2 \dots \hat{p}_n\}$. The set of the k first disks is denoted $S_k = \{p_1, p_2 \dots p_k\}$.
— For an event z, $[z]$ is the indicator function of event z (value is 1 if z is true and 0 otherwise).
— In the case where S is a set disjoint unit disks, given $p \in S$, we define $D(p)$, the disk of center $\dot{p}$ and radius $|\dot{p}NN_{\dot{S}}(\dot{p})| + 1$ (thus containing and tangent to x such that $\dot{x} = NN_{\dot{S}}(\dot{p})$).
— Given an instance $\hat{S}$, we also define $W(q) = \{p \in S; \hat{q} \in D(p)\}$.

3 Algorithm

The aim is to preprocess the set of disks S, to be able to compute $DT_{\hat{S}}$ for an instance $\hat{S}$ and the following algorithm is quite simple.

The main idea is to compute $DT_{\dot{S}}$ incrementally, determining after the insertion of $\dot{p}$ a point $\dot{x}$ that would have been a good starting point to locate $\dot{p}$ by walking in the triangulation (which is easy to find afterwards). Then when processing an instance, we use $\hat{x}$ as a starting point to locate $\hat{p}$.

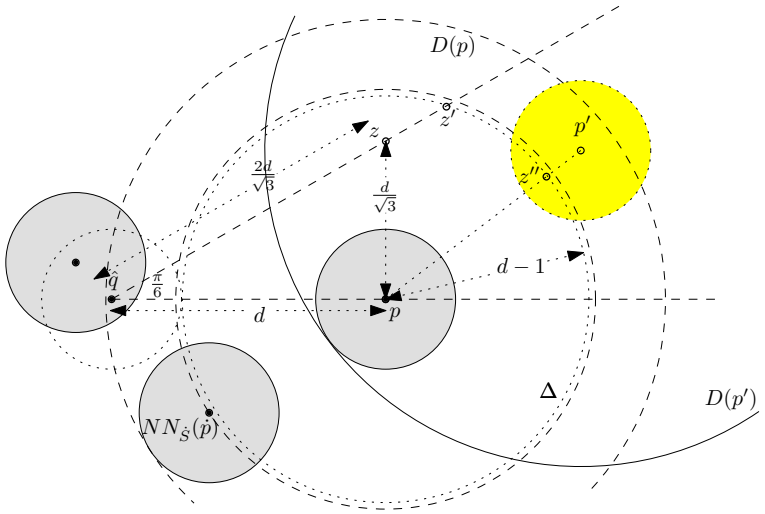

Figure 1: For the proof of Lemma 1. $D(p')$ cannot intersect q.

Preprocessing

First we assume that the indices in $S = \{p_1, p_2, \dots p_n\}$ enumerate the disks in a random order (otherwise reorder the disk according to a random permutation).

We compute $DT_{\dot{S}}$ incrementally, inserting the points in order. Furthermore after inserting $\dot{p}_k$ we compute the index $h(k)$ such that $NN_{\dot{S}_k}(\dot{p}_k) = \dot{p}_{h(k)}$. Index $h(k)$ is called the *hint for* p_k. Using incremental randomized construction, this can be done in $O(n \log n)$ expected time [8].

Instance processing

Now given an instance $\hat{S} = \{\hat{p}_i \in p_i\}$, we compute $DT_{\hat{S}}$ incrementally. The location of $\hat{p}_k$ in $DT_{\hat{S}_k}$ being done by a straight walk starting at $\hat{p}_{h(k)}$.

4 Complexity for unit disks

In this section, we assume that S is a set of disjoint unit disks. Recalling that $W(q) = \{p \in S; \hat{q} \in D(p)\}$, we start by a lemma that bound the size of $W(q)$.

Lemma 1. *$|W(q)|$ is less than 22.*

Proof. Referring to Figure 1, consider $q \in S$ and a disk p such that $\hat{q} \in D(p)$. We construct the wedge of apex $\hat{q}$ limited by the horizontal ray $\hat{q}\dot{p}$ and having an angle of $\frac{\pi}{6}$. Let $d = |\hat{q}\dot{p}|$. Consider $p' \in S$ centered in the wedge, and further than p from $\hat{q}$. We will prove below that if $d \geq 2.37$, $D(p')$ cannot contain $\hat{q}$, thus we get that in a wedge of angle $\frac{\pi}{6}$ around $\hat{q}$, among the points at distance bigger than 2.37, only the closest can belong to $W(q)$.

The size of $W(q)$ is then bounded by 12 (one point per wedge) plus the maximum number of non overlapping disk at distance less than 2.37 that is $\frac{3.37^2\pi}{\pi} - 1 \leq 10.36$ by a simple area argument (disks with center at less than

2.37 are included in a disk of radius 3.37 of center $\hat{q}$; the "-1" is because q is counted there and do not belongs to $W(q)$). We can notice that the wedge argument does not use the fact that the disks are non overlapping, thus, if the disks overlap at most k time, we have $|W(q)| \leq 11.36\,k + 12 = O(k)$.

It remains to prove that $d \geq 2.37 \Rightarrow \hat{q} \notin D(p')$ which is done by elementary (boring) geometrical computations. Referring to Figure 1 we introduce:
Δ the disk of center $\dot{p}$ and radius $d-1$,
z be the vertical projection of $\dot{p}$ on the upper wedge boundary,
z' the right most intersection of Δ and the upper wedge boundary, and
z'' the intersection of line segment $\hat{q}\dot{p}'$ with Δ.

We make the two following remarks:
Δ cannot contain any point of $\dot{S}$ (otherwise such a point would be the nearest neighbor of p and $D(p)$ would be too small to contain $\hat{q}$).
Since $d \geq 2.37 \geq \frac{\sqrt{3}}{\sqrt{3}-1}$ we have $|\dot{p}z| = \frac{d}{\sqrt{3}} \leq d-1 = |\dot{p}z'|$ that is z' is to the right of z.

Then we have that the radius of $D(p')$ is

$$
\begin{aligned}
|\dot{p}' NN_{\dot{S}}(\dot{p}')| + 1 &\leq |\dot{p}'\dot{p}| + 1 && \text{p is further than nearest neighbor}\\
&\leq |\dot{p}'z''| + |z''\dot{p}| + 1 && \text{triangular inequality}\\
&\leq |\dot{p}'z''| + (d-1) + 1 && \text{radius of } \Delta\\
&\leq |\dot{p}'z''| + \frac{2}{\sqrt{3}}d && \text{since } d \geq 2.37\\
&\leq |\dot{p}'z''| + |z\hat{q}| && \text{definition of } z\\
&\leq |\dot{p}'z''| + |z'\hat{q}| && \text{remark } z' \text{ to the right of } z\\
&\leq |\dot{p}'z''| + |z''\hat{q}| && \text{circle of center } \hat{q} \text{ through } z' \text{ cross } \Delta \text{ in } z'\\
&\leq |\dot{p}'\hat{q}| && \text{points are collinear}
\end{aligned}
$$

and thus $\hat{q} \notin D(p')$ or equivalently $p' \notin W(q)$. □

Lemma 2. *If edge $\hat{q}\hat{q}'$ of $DT_{\hat{S}}$ intersects line segment $\hat{x}\hat{p}$, with $\dot{x} = NN_{\dot{S}}(\dot{p})$, then either $\hat{q}$ or $\hat{q}'$ belongs to $D(p)$.*

Proof. By definition of $D(p)$, it encloses p and x and thus contains $\hat{p}$ and $\hat{x}$, and we can construct a circle $\Gamma \subset D(p)$ passing through $\hat{p}$ and $\hat{x}$. Since $\hat{q}\hat{q}' \in DT_{\hat{S}}$ we have $\hat{q}\hat{q}' \in DT_{\{\hat{q},\hat{q}',\hat{x},\hat{p}\}}$, that is $\hat{p}\hat{x}$ the other diagonal of the convex quadrilateral $\hat{q},\hat{x},\hat{q}'\hat{p}$ is not an edge of $DT_{\{\hat{q}\hat{q}'\hat{x}\hat{p}\}}$ or in other words a circle trough $\hat{p}$ and $\hat{x}$ cannot be empty of points of $\{\hat{q}\hat{q}'\hat{x}\hat{p}\}$. $\Gamma \subset D(p)$ is such a circle, thus it contains $\hat{q}$ or $\hat{q}'$. □

Theorem 3. *The expected cost of constructing $DT_{\hat{S}}$ is linear.*

Proof. By usual backward analysis, it is enough to prove that the insertion of the last point $\hat{p}$ is done in expected constant time.

Let $\hat{x}$ be the starting point of the straight walk in $DT_{\hat{S}\setminus\{\hat{p}\}}$ to insert $\hat{p}$. Has seen in the algorithm description, we have $\dot{x} = NN_{\dot{S}}(\dot{p})$.

The cost of locating and inserting $\hat{p}$ is split in three parts:
- the cost of turning around $\hat{x}$ in $DT_{\hat{S}\setminus\{\hat{p}\}}$,
- the cost of visiting the triangles crossed by line segment $\hat{x}\hat{p}$, and
- the cost of modifying the triangulation to update $DT_{\hat{S}\setminus\{\hat{p}\}}$ into $DT_{\hat{S}}$.

The cost of turning around $\hat{x}$ is $d^{\circ}_{DT_{\hat{S}\setminus\{\hat{p}\}}}(\hat{x}) \leq d^{\circ}_{DT_{\hat{S}}}(\hat{x})$.

The cost of updating the triangulation is $d^\circ_{DT_{\hat{S}}}(\hat{p})$.

The cost of the walk is the number of edges of $DT_{\hat{S}\setminus\{\hat{p}\}}$ crossed by $\hat{x}\hat{p}$ and we distinguish between the edges that belong to $DT_{\hat{S}}$ and those that disappear in $DT_{\hat{S}}$. We bound the number of crossed edges that disappear by the total number of edges of $DT_{\hat{S}}$ that disappear in $DT_{\hat{S}\setminus\{\hat{p}\}}$, that is $d^\circ_{DT_{\hat{S}}}(\hat{p}) - 3$. By Lemma 2 , the edges of $DT_{\hat{S}}$ crossed by $\hat{x}\hat{p}$ have a vertex in $D(p)$. Altogether the cost is bounded by

$$d^\circ_{DT_{\hat{S}}}(\hat{x}) + 2 \times d^\circ_{DT_{\hat{S}}}(\hat{p}) + \sum_{\hat{q}\in D(p)} d^\circ_{DT_{\hat{S}}}(\hat{q})$$

Then the three following claims ends the proof of the theorem.

Claim *The expected degree of $\hat{p}$ in $DT_{\hat{S}}$ is less than 6*
This is just the degree of a random point in $DT_{\hat{S}}$.

Claim *The expected degree of $\hat{x}$ in $DT_{\hat{S}}$ is less 36*
Using the well known fact that the nearest neighbor graph of a point set has bounded degree 6,

$$\begin{aligned}
E\left(d^\circ_{DT_{\hat{S}}}(\hat{x})\right) &= \frac{1}{n}\sum_{\substack{p\in S\\ \dot{x}=NN_{\dot{S}}(\dot{p})}} d^\circ_{DT_{\hat{S}}}(\hat{x}) \\
&= \frac{1}{n}\sum_{p\in S}\sum_{x\in S}[\dot{x} = NN_{\dot{S}}(\dot{p})]d^\circ_{DT_{\hat{S}}}(\hat{x}) \\
&= \frac{1}{n}\sum_{x\in S} d^\circ_{DT_{\hat{S}}}(\hat{x})\sum_{p\in S}[\dot{x} = NN_{\dot{S}}(\dot{p})] \\
&= \frac{1}{n}\sum_{x\in S} d^\circ_{DT_{\hat{S}}}(\hat{x}) d^\circ_{NN_{\dot{S}}}(\dot{x}) \\
&\leq \frac{6}{n}\sum_{x\in S} d^\circ_{DT_{\hat{S}}}(\hat{x}) \leq \frac{6}{n}\,6n = 36.
\end{aligned}$$

Claim *The expected value of $\sum_{\hat{q}\in D(p)} d^\circ_{DT_{\hat{S}}}(\hat{q})$ is less than 132*

$$\begin{aligned}
E\left(\sum_{\hat{q}\in D(p)} d^\circ_{DT_{\hat{S}}}(\hat{q})\right) &= \frac{1}{n}\sum_{p\in S}\sum_{\hat{q}\in D(p)} d^\circ_{DT_{\hat{S}}}(\hat{q}) \\
&= \frac{1}{n}\sum_{p\in S}\sum_{q\in S}[\hat{q}\in D(p)]d^\circ_{DT_{\hat{S}}}(\hat{q}) \\
&= \frac{1}{n}\sum_{q\in S}\sum_{p\in S}[p\in W(q)]d^\circ_{DT_{\hat{S}}}(\hat{q}) \\
&= \frac{1}{n}\sum_{q\in S}|W(q)|d^\circ_{DT_{\hat{S}}}(\hat{q}) \qquad \text{then by Lemma 1} \\
&\leq \frac{22}{n}\sum_{q\in S} d^\circ_{DT_{\hat{S}}}(\hat{q}) \leq \frac{22}{n}\,6n \leq 132.
\end{aligned}$$

□

5 Experiments

Point set

The set contains n disjoint imprecise points in a $2\sqrt{n} \times 2\sqrt{n}$ square. Each center of imprecise point is generated at random in the square, then the point is kept if it is at distance greater than 1 from previously kept points. Points are generated until a set of n points is obtained. A point instance is generated at random in each imprecise point.

Ten point sets of each size have been generated; the size is ranging between 10^3 and 10^7 points.

Platform

Experiments have been done on a 2.33 GHz processor with 16 GByte RAM Operating system is Linux-FC10 with CGAL3.5. Code was compiled with gcc 4.3.2 in release mode and time has been obtained with the `CGAL::Timer`.

Results

The first observation is that the number of triangles visited during the walk to locate a point from its hint is actually a small constant,. This number stays around 2.8, while for a walk between two points using spatial sort, this number of visited triangles is about 3.7.

Despite this encouraging result, our first timings were disappointing since the hint location reveals a non constant insertion time as shown by the first lines of following table (each result is averaged over 10 trials).

Since each insertion is closed to the previous one, spatial sort is less demanding for the cache memory, and this fact explains the difference in timings. Thus we experiment the following : we use spatial sort to order the imprecise points, then we preprocess the imprecise points in that order (and of course introduce the instance points in the same order), then we benefit of the locality for cache effects and of the cheap location using the hint. Processing an instance save 30% of the time compared to spatial sort.

	running time (μs) per point ♯ visited triangles per point				
n	10^3	10^4	10^5	10^6	10^7
locate from hint in random order	1.0 2.89	1.1 2.82	2.24 2.77	3.62 2.75	4.61 2.76
spatial sort + insert in order	0.2+1.2 3.76	0.2+1.2 3.62	0.25+1.20 3.71	0.37+1.19 3.67	0.56+1.22 3.55
Delaunay hierarchy	2.5 21	3.3 28	5.25 30	8.78 38	12.8 45
locate from hint in spatial sort order	1.0 2.86	1.1 2.80	1.11 2.77	1.11 2.76	1.12 2.75

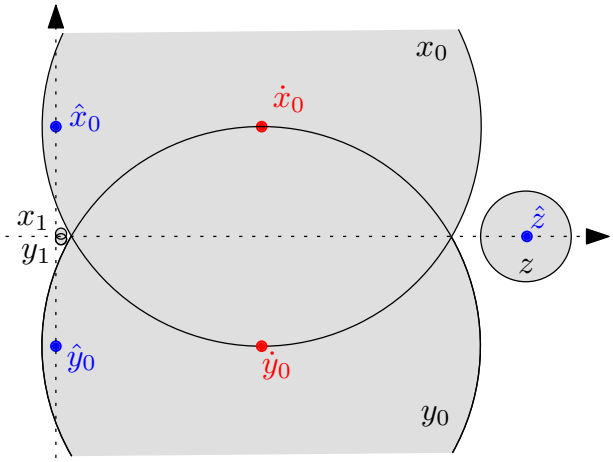

Figure 2: For disks of different radii with at most two overlapping, the complexity is $\Omega(n^2)$.

6 Beyond disjoint unit disks in the plane

6.1 Overlapping unit disks

The algorithm does not need that the disks are disjoints and have unit radius, these hypotheses are only useful for Lemma 1. By a straightforward modification of the analysis we get that n unit disks that overlap at most k times, can be preprocessed in $O(n \log n)$ time such that the Delaunay triangulation of an instance is computed $O(kn)$ time.

6.2 Non unit disks

Unfortunately, if the disks have different radii, Lemma 1 does not generalize. We have a counter example that if we allow at most two disks to overlap, the complexity of our algorithm becomes quadratic. Consider the following example of $2n+1$ disks (see Figure 2): x_i is centered in $10^{-i}(2,1)$ and has radius $10^{-i}2.1$ for $0 \leq i < n$ and $\hat{x}_i$ is placed at $10^{-i}(2,\epsilon^i)$; y_i is symmetric of x_i with respect to x-axis and z is on the right on the x-axis with $\hat{z} = \dot{z}$. Then the points are inserted in a random order (unrelated to the order of index i), to construct $DT_{\dot{S}}$. When inserting $\dot{x}_i$ its nearest neighbor is its final nearest neighbor $\dot{y}_i$ with probability $\frac{1}{2}$. Now when the instance is processed, $DT_{\hat{S}}$ links $\hat{z}$ to all other points and the location of x_i starts from y_i, with probability $\frac{1}{2}$, and crosses all edges zx_j, zy_k for $j,k \geq i$ if theses points are already inserted. Thus with constant probability insertion of a point takes linear time. If the disks have different sizes and do not overlap, the complexity of our algorithm remains open.

6.3 Higher dimension

Higher dimensions generalization needs additional hypotheses on the data, that are usual for random incremental construction. We get: if S is such that for a random sample R of size r the expected sizes of $DT_{\dot{R}}$ and $DT_{\hat{R}}$ are both

$O(r)$, then S can be preprocessed in $O(n \log n)$ time such that the Delaunay triangulation of an instance is computed $O(n)$ time.

Acknowledgments Author thanks Sylvain Lazard for fruitful discussions.

References

[1] Nina Amenta, Sunghee Choi, and Günter Rote. Incremental constructions con brio. In *Proc. 19th Annu. Sympos. Comput. Geom.*, pages 211–219, 2003.

[2] Kevin Buchin. Constructing delaunay triangulations along space-filling curves. In *Proc. 17th European Symposium on Algorithms*, volume 5757 of *Lecture Notes Comput. Sci.*, pages 119–130. Springer-Verlag, 2009.

[3] Kevin Buchin, Maarten Löffler, Pat Morin, and Wolfgang Mulzer. Delaunay triangulation of imprecise points simplified and extended. In *Proc. 18th Workshop Algorithms Data Struct.*, volume 5664 of *Lecture Notes Comput. Sci.*, pages 131–143. Springer-Verlag, 2009.

[4] Bernard Chazelle, Olivier Devillers, Ferran Hurtado, Mercè Mora, Vera Sacristán, and Monique Teillaud. Splitting a Delaunay triangulation in linear time. *Algorithmica*, 34:39–46, 2002.

[5] F. Chin and C. A. Wang. Finding the constrained Delaunay triangulation and constrained Voronoi diagram of a simple polygon in linear time. *SIAM J. Comput.*, 28:471–486, 1998.

[6] Pedro Machado Manhães de Castro and Olivier Devillers. Self-adapting point location. Research Report 7132, INRIA, 2009.

[7] Christophe Delage. Spatial sorting. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 3.5 edition, 2009.

[8] Olivier Devillers. The Delaunay hierarchy. *Internat. J. Found. Comput. Sci.*, 13:163–180, 2002.

[9] Olivier Devillers, Sylvain Pion, and Monique Teillaud. Walking in a triangulation. *Internat. J. Found. Comput. Sci.*, 13:181–199, 2002.

[10] Luc Devroye, Christophe Lemaire, and Jean-Michel Moreau. Expected time analysis for Delaunay point location. *Comput. Geom. Theory Appl.*, 29:61–89, 2004.

[11] Luc Devroye, Ernst Peter Mücke, and Binhai Zhu. A note on point location in Delaunay triangulations of random points. *Algorithmica*, 22:477–482, 1998.

[12] Leonidas J. Guibas, D. Salesin, and J. Stolfi. Epsilon geometry: building robust algorithms from imprecise computations. In *Proc. 5th Annu. Sympos. Comput. Geom.*, pages 208–217, 1989.

[13] Leonidas J. Guibas, D. Salesin, and J. Stolfi. Constructing strongly convex approximate hulls with inaccurate primitives. *Algorithmica*, 9:534–560, 1993.

[14] Maarten Löffler and Jack Snoeyink. Delaunay triangulation of imprecise points in linear time after preprocessing. *Computational Geometry*, 43:234–242, 2009.

[15] Takayuki Nagai, Seigo Yasutome, and Nobuki Tokura. Convex hull problem with imprecise input. In *japanese Conference on Discrete and Computational Geometry*, volume 1763 of *Lecture Notes in Computer Science*, pages 207–219. Springer-Verlag, 2004.

[16] R. Seidel. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Comput. Geom. Theory Appl.*, 1(1):51–64, 1991.

[17] Mariette Yvinec. 2D triangulations. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 3.5 edition, 2009.

Centre de recherche INRIA Sophia Antipolis – Méditerranée
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex

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