Delaunay triangulations,
theory vs practice.

Olivier Devillers

INRIA Sophia Antipolis - Méditerranée

Thirty years ago, at the early ages of computational geometry, the game of computational geometers was to design fancy algorithms with optimal theoretical complexities. The result was usually an algorithmic *journal article*, but not an *implementation*.

In the same period, some people were actually coding geometric algorithms, but without regard for the asymptotic complexities, and without proof of correctness of the result. They were not using the algorithms designed by theoreticians for two reasons [14]:
- these algorithms were too intricate and
- they rely on the arithmetic of real numbers, which differs from the floating-point arithmetic of computers.

These drawbacks of old computational geometry algorithms have been addressed in various ways to obtain algorithms that reconcile robustness, practical efficiency, and theoretical guarantees.

The aim of this talk is to illustrate some steps of this transition from theoretical stuff to efficient algorithms actually used in industrial applications. I will do this using my favorite problem: the construction of the Delaunay triangulation of a set of points and the dynamic maintenance of such a triangulation under point insertions and deletions.

Code used for the benchmarks is available at : `http://www.inria.fr/sophia/members/Olivier.Devillers/EuroCG2012/`.

CGAL benchmarks

To give an idea of the current performance of software computing Delaunay triangulation, we give the following timings obtained on a 16GByte, 2.3GHz workstation with CGAL 3.9 (compiled in release mode) using "Exact predicates inexact constructions kernel".

In the following table, "static insert" uses spatial sorting, "dynamic insert" uses the Delaunay hierarchy, and "deletion" removes all points in a random order.

Although theory claims that the deletion time does not depend on the triangulation size, we observe the contrary, especially in two dimensions. By profiling the code, we can observe that this phenomenon is due the load of the cache memory.

$\sharp$ points	static insert	dynamic insert	delete	maximal $\sharp$ points before swap
	μs per point			
2D				
100K	0.90	2.8	1.7	
1M	0.92	5.8	2.5	
10M	1.06	9.0	3.0	
100M	1.15	13	3.2	
230M	1.2			
swap				230M
3D				
100K	7.8	18	102	
1M	8	25	106	
10M	8.2	33	109	
swap				60M

To evaluate the cost of a correct treatment of robustness issues, we compute the Delaunay triangulation of 10 millons points using two different kernels. The "Exact predicates inexact constructions kernel" correctly handles rounding errors and degenerate configurations while the `Simple_cartesian<double>` kernel runs a little bit faster but may fail to terminate, especially on degenerate or near degenerate input.

	2D	3D
Exact predicates inexact constructions kernel	10.6 s	82 s
`Simple_cartesian` `<double>` kernel	9.7 s	75 s

Bibliographical notes

First algorithms for Delaunay triangulation appeared in the seventies.[1] The gift wrapping algorithm was proposed in 1970 to compute 2D Delaunay triangulation by Frederick, Wong, and Edge [31] and by Chand and Kapur for convex hull in 3D [12]. Incremental algorithms were introduced by Lawson in 2D in 1977 [37] and in 3D by Bowyer and Watson in 1981 [10, 44]. A gift wrapping 3D triangulation algorithm was proposed by Nguyen [40].

Regarding **worst-case optimal** algorithms, a divide-and-conquer approach was developed by Lee in 2D in 1978 [38] and Fortune proposed his plane sweep algorithm in 1986 [30]. For higher dimensions, the final solution appeared in 1993 with Chazelle's optimal algorithm for convex hull [13].

On the way to make algorithms easier to code, **randomization** was a very useful ingredient. Randomization was introduced in computational geometry by Clakson and Shor paper in 1989 [16]; from then, randomized incremental constructions (RIC) were widely used in the domain. In fact a RIC is actually a deterministic incremental algorithm but analyzed under the hypothesis of a random order for data insertion. Previously, Boissonnat and Teillaud had introduced an algorithm in this spirit for Delaunay triangulation: the Delaunay tree [9] but its correct analysis [8] was actually deduced few years later, by an application of Clarkson and Shor's techniques. A variant of the Delaunay tree was also proposed by Guibas, Knuth, and Sharir [35]. In 1992, Boissonnat, Devillers, Schott, Teillaud and Yvinec proposed the history graph (or influence graph) to allow insertions and deletions within RIC [7]. In 2002, Devillers used the Delaunay hierarchy [18] to reconcile a proven randomized complexity with good constants for both time and space.

One of the obstacles to the use of computational geometry algorithms by practitioners was their lack of **robustness** when using floating point arithmetic, since their proofs of correctness rely on the arithmetic of real numbers. One way to cope with the problem, as proposed by Sugihara and Iri [43], is to perform additional *combinatorial and topological* tests to enforce this correctness; this approach is used in Held's software VRONI [36]. With the exact computation paradigm [45] Yap proposed another solution that allows the use of algorithms developed with real arithmetic in mind: the basic geometric tests, called *predicates*, are carefully isolated in the algorithm and their evaluation is done exactly; efficient predicates for Delaunay triangulations were proposed by Shewchuk [42] and Devillers and Pion [22]. The use of the exact computation paradigm can be used only in the sensitive call to the predicate: this is the idea of structural filtering [32].

Close to robustness issues are the treatment of **degeneracies**. Symbolic perturbations are the generic answer to that problem [27, 28, 41] and exist in several variations in the literature, including special versions for Delaunay triangulations [2, 24].

Beyond the global algorithmic choices such as RIC, working on details of the algorithm design have an influence on the **constant factors** of the complexity. Many point location algorithms are implemented by walking in the triangulation, which can be done in several ways [23, 17]. Depending on the size of the point set, the location strategy can be adapted from a brute force search for very small point sets, to a walk, or jump & walk [39, 25], or Delaunay hierarchy [18] when the size increases. If the points are known in advance, a true random order have some disadvantage and *spatial sort* can be used for preprocessing [4, 11].

Deleting one point of degree d from a 2D Delaunay triangulation can be done in $O(d)$ time, by using a quite complicated algorithm originally designed for the Delaunay triangulation of a convex polygon by Aggarwal, Guibas, Saxe, and Shor [1]. A much simpler randomized $O(d)$ solution has been proposed by Chew [15]. In practice, various non-optimal solutions of complexity $O(d \log d)$ or $O(d^2)$ are often preferred since d is usually a small number [19]. A specialized implementation for small value of d, optimizing the number of incircle tests and the memory management can yield substantial improvement [20]. In 3D or higher dimensions, a way of managing point deletion is to compute from scratch the (small) Delaunay triangulation of the neighbors of the removed point, and to plug the relevant simplices inside the hole created in the whole triangulation by the removal of the simplices incident to the deleted point.

In dimension 3 or higher, the **size of the Delaunay** triangulation is not constrained by the number of points as it is in 2D. There are several results about that size under various hypotheses. For random points in a ball in any fixed dimension, Dwyer proved that the expected size is $\Theta(n)$ [26]. For random points in 3D on the boundary of a polyhedron, the expected size was proved to be $\Theta(n)$ [34] in the convex case and between $\Omega(n)$ and $O(n \log n)$ [33] in the non convex case by Golin and Na. If the probabilistic hypothesis is replaced by some hypotheses controlling the point distribution on the boundary of the polyhedron (called (ϵ, κ)-sampling) then the size of the triangulation has been proved to be $\Theta(n)$ [5] by

[1] thanks to Jonathan Shewchuk for his summary on this topic in the compgeom mailing list.

Attali and Boissonnat. The same authors with Lieutier proved that, applying this sampling hypotheses to a generic smooth surface, yields an $O(n \log n)$ complexity [6]. If the smooth surface is non generic (e.g. a cylinder) and sampled with the same hypotheses, the complexity can be $\Omega(n\sqrt{n})$ [29] as proved by a construction of Erickson. Erickson, Devillers, and Goaoc proved that the triangulation of randomly distributed points on a cylinder has complexity $\Theta(n \log n)$ [21]. In higher dimensions, Amenta, Attali, and Devillers proved that a good-sampling of a p-dimensional polyhedron embedded in dimension d has $O(n^k)$ size with $k = \frac{d+1-\lceil \frac{d+1}{p+1} \rceil}{p}$ [3].

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 Comput. Geom.*, 4(6):591–604, 1989. http://www.springerlink.com/content/7722150064q64865/.

[2] Pierre Alliez, Olivier Devillers, and Jack Snoeyink. Removing degeneracies by perturbing the problem or the world. *Reliable Computing*, 6:61–79, 2000. Special Issue on Computational Geometry, http://hal.inria.fr/inria-00338566.

[3] Nina Amenta, Dominique Attali, and Olivier Devillers. A tight bound for the Delaunay triangulation of points on a polyhedron. Research Report 6522, INRIA, 2008. http://hal.inria.fr/inria-00277899.

[4] Nina Amenta, Sunghee Choi, and Günter Rote. Incremental constructions con BRIO. In *Proc. 19th Annu. Sympos. Comput. Geom.*, pages 211–219, 2003. http://page.inf.fu-berlin.de/~rote/Papers/pdf/Incremental+constructions+con+BRIO.pdf.

[5] Dominique Attali and Jean-Daniel Boissonnat. A linear bound on the complexity of the Delaunay triangulation of points on polyhedral surfaces. *Discrete and Computational Geometry*, 31:369–384, 2004. http://www.springerlink.com/content/udhapea67abmq086/.

[6] Dominique Attali, Jean-Daniel Boissonnat, and André Lieutier. Complexity of the Delaunay triangulation of points on surfaces: The smooth case. In *Proc. 19th Annual Symposium on Computational Geometry*, pages 237–246, 2003. http://dl.acm.org/citation.cfm?id=777823.

[7] J.-D. Boissonnat, O. Devillers, R. Schott, M. Teillaud, and M. Yvinec. Applications of random sampling to on-line algorithms in computational geometry. *Discrete and Computational Geometry*, 8:51–71, 1992. http://hal.inria.fr/inria-00090675.

[8] J.-D. Boissonnat and M. Teillaud. On the randomized construction of the Delaunay tree. *Theoret. Comput. Sci.*, 112:339–354, 1993. http://www.sciencedirect.com/science/article/pii/030439759390024N.

[9] Jean-Daniel Boissonnat and Monique Teillaud. A hierarchical representation of objects: The Delaunay tree. In *Proc. 2nd Annu. Sympos. Comput. Geom.*, pages 260–268, 1986. http://dl.acm.org/citation.cfm?id=10543.

[10] A. Bowyer. Computing Dirichlet tesselations. *Comput. J.*, 24:162–166, 1981. http://comjnl.oxfordjournals.org/content/24/2/162.short.

[11] 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. http://www.springerlink.com/index/m17216w072m54438.pdf.

[12] D. R. Chand and S. S. Kapur. An algorithm for convex polytopes. *J. ACM*, 17(1):78–86, January 1970. http://dl.acm.org/citation.cfm?id=321564.

[13] Bernard Chazelle. An optimal convex hull algorithm in any fixed dimension. *Discrete Comput. Geom.*, 10:377–409, 1993. http://www.springerlink.com/content/f62210744m187220/.

[14] Bernard Chazelle et al. Application challenges to computational geometry: CG impact task force report. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, volume 223 of *Contemporary Mathematics*, pages 407–463. American Mathematical Society, Providence, 1999.

[15] L. P. Chew. Building Voronoi diagrams for convex polygons in linear expected time. Technical Report PCS-TR90-147, Dept. Math. Comput. Sci., Dartmouth College, Hanover, NH, 1990. http://www.cs.dartmouth.edu/reports/TR90-147.pdf.

[16] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989. http://www.springerlink.com/content/b9n24vr730825p71/.

[17] Pedro Machado Manhães de Castro and Olivier Devillers. Simple and efficient distribution-sensitive point location, in triangulations. In *Workshop on Algorithm Engineering and Experiments*, pages 127–138, 2011. http://www.siam.org/proceedings/alenex/2011/alenex11.php.

[18] Olivier Devillers. The Delaunay hierarchy. *Internat. J. Found. Comput. Sci.*, 13:163–180, 2002. http://hal.inria.fr/inria-00166711.

[19] Olivier Devillers. On deletion in Delaunay triangulation. *Internat. J. Comput. Geom. Appl.*, 12:193–205, 2002. http://hal.inria.fr/inria-00167201.

[20] Olivier Devillers. Vertex removal in two dimensional Delaunay triangulation: Speed-up by low degrees optimization. *Computational Geometry: Theory and Applications*, 44:169–177, 2011. http://hal.inria.fr/inria-00560379/.

[21] Olivier Devillers, Jeff Erickson, and Xavier Goaoc. Empty-ellipse graphs. In *Proc. 19th ACM-SIAM Sympos. Discrete Algorithms*, pages 1249–1256, 2008. http://hal.inria.fr/inria-00176204.

[22] Olivier Devillers and Sylvain Pion. Efficient exact geometric predicates for Delaunay triangulations. In *Proc. 5th Workshop Algorithm Eng. Exper.*, pages 37–44, 2003. http://hal.inria.fr/inria-00344517/.

[23] Olivier Devillers, Sylvain Pion, and Monique Teillaud. Walking in a triangulation. *Internat. J. Found. Comput. Sci.*, 13:181–199, 2002. http://hal.inria.fr/inria-00102194/.

[24] Olivier Devillers and Monique Teillaud. Perturbations for Delaunay and weighted Delaunay 3d triangulations. *Computational Geometry: Theory and Applications*, 44:160–168, 2011. http://hal.archives-ouvertes.fr/inria-00560388/.

[25] Luc Devroye, Ernst Peter Mücke, and Binhai Zhu. A note on point location in Delaunay triangulations of random points. *Algorithmica*, 22:477–482, 1998. http://cg.scs.carleton.ca/~luc/devroye_mucke_zhu_1998_a_note_on_point_location_in_delaunay_triangulations_of_random_points.pdf.

[26] R. Dwyer. The expected number of k-faces of a Voronoi diagram. *Internat. J. Comput. Math.*, 26(5):13–21, 1993. http://www.sciencedirect.com/science/article/pii/0898122193900687.

[27] H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graph.*, 9(1):66–104, 1990. http://dl.acm.org/citation.cfm?id=77639.

[28] I. Emiris and J. Canny. A general approach to removing degeneracies. *SIAM J. Comput.*, 24:650–664, 1995. http://epubs.siam.org/sicomp/resource/1/smjcat/v24/i3/p650_s1.

[29] Jeff Erickson. Dense point sets have sparse Delaunay triangulations or "...but not too nasty". *Discrete & Computational Geometry*, 33:83–115, 2005. http://www.springerlink.com/content/91ukk7qdwvpx77d1/.

[30] S. J. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2:153–174, 1987. http://www.springerlink.com/content/n88186tl165168rw/.

[31] C. O. Frederick, Y. C. Wong, and F. W. Edge. Two-dimensional automatic mesh generation for structural analysis. *Internat. J. Numer. Methods Eng.*, 2:133–144, 1970. http://onlinelibrary.wiley.com/doi/10.1002/nme.1620020112/abstract.

[32] Stefan Funke, Kurt Mehlhorn, and Stefan Näher. Structural filtering: A paradigm for efficient and exact geometric programs. *Comput. Geom. Theory and Appl.*, pages 179–194, 2005. http://www.sciencedirect.com/science/article/pii/S0925772104001348.

[33] Mordecai J. Golin and Hyeon-Suk Na. The probabilistic complexity of the voronoi diagram of points on a polyhedron. In *Proc. 18th Annual Symposium on Computational Geometry*, 2002. http://www.cse.ust.hk/~golin/pubs/SCG_02.pdf.

[34] Mordecai J. Golin and Hyeon-Suk Na. On the average complexity of 3d-voronoi diagrams of random points on convex polytopes. *Computational Geometry: Theory and Applications*, 25:197–231, 2003. http://www.cse.ust.hk/~golin/pubs/3D_Voronoi_I.pdf.

[35] Leonidas J. Guibas, D. E. Knuth, and Micha Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, 7:381–413, 1992. http://www.springerlink.com/content/p8377h68j8216860/.

[36] M. Held. Vroni: An engineering approach to the reliable and efficient computation of Voronoi diagrams of points and line segments. *Comput. Geom. Theory Appl.*, 18:95–123, 2001. http://www.sciencedirect.com/science/article/pii/S0925772101000037.

[37] C. L. Lawson. Software for C^1 surface interpolation. In J. R. Rice, editor, *Math. Software III*, pages 161–194. Academic Press, New York, NY, 1977. http://www.mendeley.com/research/software-for-c1-interpolation/.

[38] D. T. Lee. *Proximity and reachability in the plane.* PhD thesis, Coordinated Science Lab., Univ. Illinois, Urbana, Ill., 1978. http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA069764.

[39] 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. Sympos. Comput. Geom.*, pages 274–283, 1996. http://dl.acm.org/citation.cfm?id=237396.

[40] V. Ph. Nguyen. Automatic mesh generation with tetrahedron elements. *Internat. J. Numer. Methods Eng.*, 18:273–289, 1982.

[41] R. Seidel. The nature and meaning of perturbations in geometric computing. *Discrete Comput. Geom.*, 19:1–17, 1998. http://www.springerlink.com/content/px52xh005cxxvdku/.

[42] Jonathan Richard Shewchuk. Adaptive precision floating-point arithmetic and fast robust geometric predicates. *Discrete Comput. Geom.*, 18(3):305–363, 1997. http://www.springerlink.com/content/gfepd5qjykp9mk1f/.

[43] K. Sugihara and M. Iri. Construction of the Voronoi diagram for 'one million' generators in single-precision arithmetic. *Proc. IEEE*, 80(9):1471–1484, September 1992. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=163412&tag=1.

[44] D. F. Watson. Computing the n-dimensional Delaunay tesselation with applications to Voronoi polytopes. *Comput. J.*, 24(2):167–172, 1981. http://comjnl.oxfordjournals.org/content/24/2/167.short.

[45] C. K. Yap and T. Dubé. The exact computation paradigm. In D.-Z. Du and F. K. Hwang, editors, *Computing in Euclidean Geometry*, volume 4 of *Lecture Notes Series on Computing*, pages 452–492. World Scientific, Singapore, 2nd edition, 1995. http://cs.nyu.edu/cs/faculty/yap/papers/paradigm.ps.