# Delaunay triangulation and randomized constructions

Olivier Devillers

| | |
|---|---|
| Title: | Delaunay triangulation and randomized constructions |
| Name: | Olivier Devillers[1] |
| Affil./Addr. | INRIA, France |
| Keywords: | Delaunay triangulation; Voronoi diagram; Randomization; Convex hull |
| SumOriWork: | 1989; Clarkson, Shor<br>1993; Seidel<br>2002; Devillers<br>2003;Amenta, Choi, Rote |

# Delaunay triangulation and randomized constructions

Olivier Devillers[1]
INRIA, France

## Years aud Authors of Summarized Original Work

1989; Clarkson, Shor
1993; Seidel
2002; Devillers
2003;Amenta, Choi, Rote

## Keywords

Delaunay triangulation; Voronoi diagram; Randomization; Convex hull

## Problem Definition

The Delaunay triangulation and the Voronoi diagram are two classic geometric structures in the field of computational geometry. Their success can perhaps be attributed to two main reasons: Firstly, there exist practical, efficient algorithms to construct them; and secondly, they have an enormous number of useful applications ranging from meshing and 3D-reconstruction to interpolation.

Given a set $S$ of $n$ sites in some space $\mathbb{E}$, we define the Voronoi region$V_S(p)$, of $p \in S$ to be the set of points in $\mathbb{E}$ whose nearest neighbor in $S$ is $p$ (for some distance $\delta$):

$$V(p) = \left\{ x \in \mathbb{E}, \forall q \in S \backslash \{p\} \;\; \delta(x,p) < \delta(x,q) \right\}.$$

It is easily seen that these regions form a partition of $\mathbb{E}$ into convex regions which we refer to as *cells*. These concepts may be extended into more exotic spaces such as periodic and hyperbolic spaces or metric spaces using convex distances, though we restrict ourselves to here to the case where $\mathbb{E}$ is the Euclidean space $\mathbb{E} = \mathbb{R}^d$ and the distance $\delta$ is the $L_2$ norm.

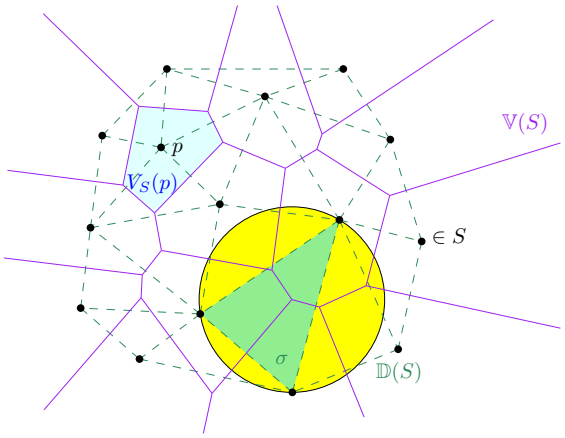


**Fig. 1.** The Voronoi diagram of a set $S$ of 15 points and its dual Delaunay triangulation

The Voronoi diagram $\mathbb{V}(S)$ may now be defined as the limit between the different Voronoi cells

$$\mathbb{V}(S) = \mathbb{E} \setminus \bigcup_{p \in S} V_S(p).$$

The Delaunay triangulation $\mathbb{D}(S)$ is the geometric dual of $\mathbb{V}(S)$. More formally, $\mathbb{D}(S)$ is a simplicial complex defined by:

$$\sigma \in \mathbb{D}(S) \Longleftrightarrow \bigcap_{p \in \sigma} \overline{V_S}(p) \neq \emptyset,$$

where $\overline{V_S}(p)$ is the closure of the Voronoi cell $V_S(p)$ (see Figure 1).

Voronoi diagrams and Delaunay triangulations have received a lot of attention in the literature with several surveys, books, or book chapters (e.g. [4; 14]) and hundreds of papers. In this article, we will focus on randomized construction algorithms for the Delaunay triangulation. Such algorithms use randomness to speed up their running time but do not assume any randomnsess in the data distribution.

# Key Results

## Delaunay Properties

***Empty Ball Property*** One crucial property of the Delaunay triangulation, which is the basis of many algorithms is the *empty ball property*, which guarantees that a triangle is a Delaunay triangle of $S$ if and only if the interior its circumball does not contain any point of $S$.

***Size of the Triangulation*** In the plane, the combinatorial properties of a triangulation (not necessarily Delaunay) are completely fixed, by the Euler relation. In particular, given $n$ vertices, $h$ of which on the convex hull, every triangulation must have $2n-2-h$ triangles and $3n-3-h$ edges. In dimension $d$, the Dehn-Sommerville relations yield a linear dependence for the number of simplices of all dimensions on the number of simplices of dimensions $k$ for $k \leq \lceil \frac{d}{2} \rceil$; this gives an $O\left(n^{\lceil \frac{d}{2} \rceil}\right)$ upper bound for the number of simplices of all dimension. For both Delaunay and more general triangulations, these bounds are tight in the worst case.

These bounds can be tightened given some assumptions on the distribution of the input sites. If the points are uniformly distributed in a compact convex of fixed volume, then the triangulation size (its total number of simplices) is $\Theta(n)$, with a constant

exponential in $d$ [9]. In 3D, and for reconstruction purposes, it is convenient to assume that the points lie on a surface. It is known that the Delaunay triangulation of points uniformly distributed on a convex polyhedron has size $\Theta(n)$ (for a constant depending on the polyhedron complexity). For points uniformly distributed on a (non convex) polyhedron, the triangulation's size is between $\Omega(n)$ and $O(n \log n)$ [12]. If, instead of making a probabilistic assumption, we assume that the points are a 'good sampling' of the surface such that every small ball centered on the surface contains between 1 and $\kappa$ points (where $\kappa$ is a constant), then the size of the Delaunay triangulation is $\Theta(n)$ for a polyhedron, $O(n \log n)$ for a *generic* smooth surface [3], and $\Omega(n\sqrt{n})$ for a non generic surface (e.g. a cylinder). In the case of the cylinder, a uniformly distributed point set as a triangulation of size $\Theta(n \log n)$ In dimension $d$, a $p$-dimensional polyhedron whose faces have a 'good sampling' has size $O(n^k)$ where $k = \frac{d+1-\left\lceil \frac{d+1}{p+1} \right\rceil}{p}$ [2].

***First Algorithms*** Many classical techniques in algorithmic and computational geometry have been used to attack the problem of constructing the Delaunay triangulation and the Voronoi Diagram. The gift wrapping and the incremental approaches were introduced in the 1970s [11], followed by some worst-case optimal algorithms in 2D, based on divide-and-conquer [13] and sweep line techniques [10]. In higher dimensions, the optimal worst case construction of Delaunay triangulation and convex hulls was solved in the 1990s.

In the remainder of the paper we will describe some further algorithmic techniques that may be used to construct the Delaunay triangulation.

## Randomized Construction

One popular and efficient method, applied to the Delaunay triangulation at the end of the 1980s [5], is Randomized Incremental Construction (RIC). The idea is to exploit the simplicity of an incremental algorithm whilst avoiding its worst case behavior by simply adjusting the order of insertion of the points.

**Conflict Graph** Recalling that $\mathbb{D}(S)$ is the set of triangles with vertices in $S$ whose circumballs are empty, the idea is to maintain for a sequence $\varnothing = S_0 \subset S_1 \subset S_2 \subset \ldots \subset S_n = S$, where $|S_i| = i$ a sequence of triangulations $\mathbb{D}(S_i)$ with associated *conflict graphs*. We define the conflict graph to be a bipartite graph that links a point $p$ of $S \setminus S_i$ to a simplex $\sigma$ in $\mathbb{D}(S_i)$ if the circumball of $\sigma$ contains $p$ ($p$ *and* $\sigma$ *are called in conflict).* The information contained in the conflict graph simplifies the construction of $\mathbb{D}(S_{i+1})$ from $\mathbb{D}(S_i)$ since it gives directly the simplices in $\mathbb{D}(S_i) \setminus \mathbb{D}(S_{i+1})$.

The key point comes from an analysis based on random sampling [6], let's assume that $S_i$ is a random sample of size $i$ of $S$. We say that a simplex has *width* $j$ if it has $j$ points in conflict in $S$. In which case a Delaunay simplex is a simplex of width 0. Denote by $\Delta_j$ the number of simplices of width $j$ and let $\Delta_{\leq k} = \sum_{j \leq k} \Delta_j$. We first bound $\Delta_{\leq k}$ using the following remark: a simplex of width $j$ is a Delaunay simplex of a random sample $R$ of size $\frac{n}{k}$ of $S$ with probability $p_k = \frac{1}{k^{d+1}}(1 - \frac{1}{k})^j$ (vertices of $\sigma$ must be chosen in $R$ and points in conflict must not). Notice that for $j \in [2..k]$ we have $(1 - \frac{1}{k})^j \geq (1 - \frac{1}{k})^k \geq \frac{1}{4}$ since$(1 - \frac{1}{x})^x$ is an increasing fonction of value $\frac{1}{4}$ for $x = 2$. For $k \geq 2$, we have (using $\mathbb{P}$ to denote the probability measure):

$$|\mathbb{D}(R)| = O\left(\left(\frac{n}{k}\right)^{\left\lceil \frac{d}{2} \right\rceil}\right) = \sum_{\sigma \in S^{d+1}} \mathbb{P}(\sigma \in \mathbb{D}(R)) = \sum_{j \leq n} \Delta_j p_j \geq \frac{\Delta_0}{k^{d+1}} + \frac{\Delta_1 \frac{1}{2}}{k^{d+1}} + \sum_{j=2}^{k} \frac{\Delta_j \frac{1}{4}}{k^{d+1}} = \frac{\Delta_{\leq k}}{4k^{d+1}},$$

$$\Delta_{\leq k} \leq O\left(\left(\frac{n}{k}\right)^{\lceil \frac{d}{2} \rceil} k^{d+1}\right) = O\left(\left(n^{\lceil \frac{d}{2} \rceil} k^{\lceil \frac{d+1}{2} \rceil}\right)\right).$$

We can now analyze the incremental construction of $\mathbb{D}(S)$. The probability that a triangle of width $j$ appears during the construction is

$$p'_j = \frac{\binom{d+1}{j+d+1}}{(j+d+1)!}$$

(the number of permutations that look at the vertices of $\sigma$ before the points in conflict divided by the total number of permutations). Then the cost of the algorithm is given by the total number of conflicts occurring during the construction:

$$\sum_{\sigma \in S^{d+1}} \text{width}(\sigma)\mathbb{P}(\sigma\text{appears}) = \sum_j \Delta_j \cdot j \cdot p'_j = \sum_j (\Delta_{\leq j} - \Delta_{\leq j-1}) j \cdot p'_j$$

$$= \sum_j \Delta_{\leq j}(j \cdot p'_j - (j+1)p'_{j+1}) \leq \sum_j n^{\lceil \frac{d}{2} \rceil} j^{\lceil \frac{d+1}{2} \rceil} O\left(\frac{1}{j^{d+2}}\right) \leq O\left(n^{\lceil \frac{d}{2} \rceil} \sum_j j^{-\lceil \frac{d}{2} \rceil}\right)$$

which gives $O(n \log n)$ for $d = 2$ and $O\left(n^{\lceil \frac{d}{2} \rceil}\right)$ for higher $d$.

**Backward Analysis** A simpler way of analyzing RIC is backward analysis [15], and we will sketch it in 2D. The idea is quite simple and consists in asking: *what is the cost of the last step?* The answer is that the cost of modifying the triangulation during last insertion is clearly proportional to the *degree* (number of simplices incident) of the last point inserted into the triangulation. Since the last point is a random point, its expected degree is

$$\mathbb{E}\text{xp}(\text{degree(last point)}) = \mathbb{E}\text{xp}(\text{degree(random point)}) = \frac{1}{n}\sum_{p \in S} \text{degree}(p) \leq \frac{6n}{n} = 6,$$

and summing over all insertion steps gives a linear cost for updating the triangulation. It remains to count the cost of updating the conflict graph. We remark that there is a conflict between the last point $p_n$ and a triangle created by the insertion of the $j$th point $p_j$ if and only if the edge $p_j p_n$ exists in $\mathbb{D}(S_j \cup \{p_n\})$. Since $p_j$ and $p_n$ are both random points of $S_j \cup \{p_n\}$, it happens with probability $O\left(\frac{1}{j}\right)$, the expected number of conflicts for $p_n$ is thus $O\left(\sum_j \frac{1}{j}\right) = O(\log n)$, and the total number of conflicts is $O\left(\sum_k \log k\right) = O(n \log n)$.

**Delaunay Hierarchy** The conflict graph approach assumes complete knowledge of $S$ to initialize the conflict graph. Using a lazy approach and postponing the conflict determination it is possible to obtain on-line algorithms [5].

Amongst the on-line schemes to construct the Delaunay triangulation, the Delaunay hierarchy [7] gives good results both in theory and in practice. The Delaunay hierarchy constructs a sequence of random samples $S = S_0 \supset S_1 \supset \ldots \supset S_h$ such that $\mathbb{P}\left(p \in S_i \mid p \in S_{i-1}\right) = \alpha$. Then the Delaunay triangulations of $\mathbb{D}(S_i), 1 \leq i \leq h$ are maintained under point insertions. Pointers from a vertex of $\mathbb{D}(S_i)$ to the vertices at the same position in $\mathbb{D}(S_{i-1})$ (if $i < h$) and in $\mathbb{D}(S_{i+1})$ (if it actually belongs to $S_{i+1}$) are also computed.

When a new point $p$ needs to be inserted, it is located by walking in $\mathbb{D}(S_h)$ (using neighborhood relations) to reach the closest vertex, $w_h$ of $p$ in $\mathbb{D}(S_h)$. Then the hierarchy is descended, walking in $\mathbb{D}(S_i)$ from $w_{i+1}$ to find $w_i$ the closest neighbor in

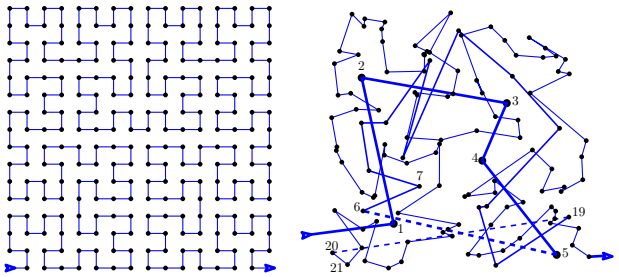


**Fig. 2.** Right: The Hilbert space filling curve. Left: random points sorted with BRIO

sample $S_i$. Using these neighbors, it is easy to insert $p$ in $\mathbb{D}(S_h)$ and in the triangulation of other samples that the random process assigns to $p$.

In 2D, the expected cost of the walk at any level is $O(\alpha^{-1})$ and the expected value for $h$ is $O\left(\frac{\log n}{-\log \alpha}\right)$. Thus the theoretical complexity of the algorithm is $O(n \log n)$. The value of $\alpha$ can be optimized depending on the input distribution: for random points $\alpha = \frac{1}{30}$ gives good timings and a very low memory requirement in addition to the one needed for $\mathbb{D}(S)$.

**A Less Randomized Construction** Constructing the Delaunay triangulation by inserting the points in a random order presents a drawback with respect to memory management. Since the inserted point is random in $S$, there is very little chance that the triangles needed are present in the cache memory. So, an idea is to sort the points using a space filling curve (see Fig. 2-left) to ensure locality of the insertions. Unfortunately, when inserting the points in such an order, the randomized complexity results no longer apply and the number of created and destroyed triangles during the construction may explode on certain data sets.

A smart solution has been proposed: it is possible to use an insertion order random enough to apply randomized complexity results and allowing some locality to benefit from cache memory. Brio (Biased randomized Insertion Order) [1] proposes to partition $S$ in a set of random samples $S = \bigcup_{0 \leq i \leq h} S_i$ such that $|S_i| = \alpha |S_{i+1}|$ for $\alpha \leq 1$ a small constant (e.g. $\alpha = \frac{1}{4}$) and to insert the samples by increasing size, each sample being sorted using a spatial filling curve (see Fig. 2-right).

In the random setting, we have seen that the probability for a triangle of width $j$ to appear in the conflict graph algorithm was $\frac{1 \cdot 2 \cdot 3}{(j+1)(j+2)(j+3)} = \Theta(j^{-}3)$. Using BRIO, this probability is a bit less intricate to compute, but it can be bounded in terms of $\alpha$ and it can be shown that it is still $\Theta(j^{-3})$ and thus randomized complexity results still apply.

# Experimental Results

On a 16GBytes, 2.3GHz desktop CGAL currently computes the Delaunay triangulation of up to 200M points in 2D and 50M points in 3D [8].

Static timings are almost constant with respect to the total number of points and are about $1\mu$s per point in 2D and $8\mu$s per point in 3D. In the dynamic setting, 1 million points are processed in 6s in 2D and 25s in 3D.

# URLs to Code and Data Sets

CGAL, amongst a big collection of computational geometry algorithms, provides implementations for Delaunay triangulations in 2D, 3D, and general dimension. It computes

the Delaunay triangulation in 2D and 3D using the Delaunay hierarchy in a dynamic setting, and using BRIO for static computation (`http://cgal.org`).

# Cross-References

—"Voronoi Diagrams and Delaunay Triangulations" entry about Fortune's algorithm in Computational geometry chapter (Fekete).
—"Compact triangulation data structures" entry in *the other (!)* Computational geometry chapter (Haverkort | L. Castelli, J. Rossignac, and O. Devillers).
— Probably several entries in the mesh chapter.
— Randomized algorithms ?

# Recommended Reading


1. Amenta N, Choi S, Rote G (2003) Incremental constructions con BRIO. In: Proc. 19th Annu. Sympos. Comput. Geom., pp 211–219, DOI 10.1145/777792.777824, URL `http://page.inf.fu-berlin.de/~rote/Papers/pdf/Incremental+constructions+con+BRIO.pdf`
2. Amenta N, Attali D, Devillers O (2012) A tight bound for the Delaunay triangulation of points on a polyhedron. Discrete & Computational Geometry 48:19–38, DOI 10.1007/s00454-012-9415-7, URL `http://hal.inria.fr/hal-00784900`
3. Attali D, Boissonnat JD, Lieutier A (2003) Complexity of the Delaunay triangulation of points on surfaces: The smooth case. In: Proc. 19th Annual Symposium on Computational Geometry, pp 201–210, DOI 10.1145/777792.777823, URL `http://dl.acm.org/citation.cfm?id=777823`
4. Aurenhammer F, Klein R (2000) Voronoi diagrams. In: Sack JR, Urrutia J (eds) Handbook of Computational Geometry, Elsevier Science Publishers B.V. North-Holland, Amsterdam, pp 201–290, URL `ftp://ftp.cis.upenn.edu/pub/cis610/public_html/ak-vd-00.ps`
5. Boissonnat JD, Teillaud M (1986) A hierarchical representation of objects: The Delaunay tree. In: Proc. 2nd Annu. Sympos. Comput. Geom., pp 260–268, URL `http://dl.acm.org/citation.cfm?id=10543`
6. Clarkson KL, Shor PW (1989) Applications of random sampling in computational geometry, II. Discrete Comput Geom 4:387–421, DOI 10.1007/BF02187740, URL `http://www.springerlink.com/content/b9n24vr730825p71/`
7. Devillers O (2002) The Delaunay hierarchy. Internat J Found Comput Sci 13:163–180, DOI 10.1142/S0129054102001035, URL `http://hal.inria.fr/inria-00166711`
8. Devillers O (2012) Delaunay triangulations, theory vs practice. In: Abstracts 28th European Workshop on Computational Geometry, pp 1–4, URL `http://hal.inria.fr/hal-00850561`, invited talk
9. Dwyer R (1993) The expected number of $k$-faces of a Voronoi diagram. Internat J Comput Math 26(5):13–21, DOI 10.1016/0898-1221(93)90068-7, URL `http://www.sciencedirect.com/science/article/pii/0898122193900687`
10. Fortune SJ (1987) A sweepline algorithm for Voronoi diagrams. Algorithmica 2:153–174, DOI 10.1007/BF01840357, URL `http://www.springerlink.com/content/n88186tl165168rw/`
11. Frederick CO, Wong YC, Edge FW (1970) Two-dimensional automatic mesh generation for structural analysis. Internat J Numer Methods Eng 2:133–144, DOI 10.1002/nme.1620020112/abstract
12. Golin MJ, Na HS (2002) The probabilistic complexity of the voronoi diagram of points on a polyhedron. In: Proc. 18th Annual Symposium on Computational Geometry, URL `http://www.cse.ust.hk/~golin/pubs/SCG_02.pdf`
13. Lee DT (1978) Proximity and reachability in the plane. PhD thesis, Coordinated Science Lab., Univ. Illinois, Urbana, Ill., URL `http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA069764`
14. Okabe A, Boots B, Sugihara K (1992) Spatial Tessellations: Concepts and Applications of Voronoi Diagrams. John Wiley & Sons, Chichester, UK
15. Seidel R (1993) Backwards analysis of randomized geometric algorithms. In: Pach J (ed) New Trends in Discrete and Computational Geometry, Algorithms and Combinatorics, vol 10, Springer-Verlag, pp 37–68, URL `http://ftp.icsi.berkeley.edu/ftp/pub/techreports/1992/tr-92-014.ps.gz`