Delaunay triangulations, theory vs practice.
Olivier Devillers

To cite this version:
Olivier Devillers. Delaunay triangulations, theory vs practice.. EuroCG, 28th European Workshop on Computational Geometry, 2012, Assisi, Italy. hal-00850561

HAL Id: hal-00850561
https://inria.hal.science/hal-00850561
Submitted on 28 Apr 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
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:

<table>
<thead>
<tr>
<th>$#$ points</th>
<th>static insert</th>
<th>dynamic insert</th>
<th>delete</th>
<th>maximal $#$ points before swap</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100K</td>
<td>0.90</td>
<td>2.8</td>
<td>1.7</td>
<td></td>
</tr>
<tr>
<td>1M</td>
<td>0.92</td>
<td>5.8</td>
<td>2.5</td>
<td></td>
</tr>
<tr>
<td>10M</td>
<td>1.06</td>
<td>9.0</td>
<td>3.0</td>
<td></td>
</tr>
<tr>
<td>100M</td>
<td>1.15</td>
<td>13</td>
<td>3.2</td>
<td></td>
</tr>
<tr>
<td>230M</td>
<td>1.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>swap</td>
<td></td>
<td></td>
<td></td>
<td>230M</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3D</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>100K</td>
<td>7.8</td>
<td>18</td>
<td>102</td>
<td></td>
</tr>
<tr>
<td>1M</td>
<td>8</td>
<td>25</td>
<td>106</td>
<td></td>
</tr>
<tr>
<td>10M</td>
<td>8.2</td>
<td>33</td>
<td>109</td>
<td></td>
</tr>
<tr>
<td>swap</td>
<td></td>
<td></td>
<td></td>
<td>60M</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>$#$ points</th>
<th>static insert</th>
<th>dynamic insert</th>
<th>delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.6 s</td>
<td>10.6 s</td>
<td>82 s</td>
<td></td>
</tr>
<tr>
<td>9.7 s</td>
<td>75 s</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To evaluate the cost of a correct treatment of robustness issues, we compute the Delaunay triangulation of 10 millions 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.
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, 11]. 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 [35] and Fortune proposed his plane sweep algorithm in 1986 [39]. 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 Clarkson and Shor paper in 1989 [15]; 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 [2] 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 [33], 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 [41], 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 [11, 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 Θ(n) [24]. For random points in 3D on the boundary of a polyhedron, the expected size was proved to be Θ(n) [34] in the convex case and between Ω(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 Θ(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 $\Omega(\sqrt{n})$ [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 $k = \frac{d+1-\lceil \frac{d+1}{p} \rceil}{p}$ [3].

References


