Robust and Efficient Delaunay triangulations of points on or close to a sphere

Manuel Caroli, Pedro Machado Manhães de Castro, Sebastien Loriot, Camille Wormser, Monique Teillaud

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Robust and Efficient Delaunay triangulations of points on or close to a sphere

Manuel Caroli — Pedro M. M. de Castro — Sébastien Loriot — Camille Wormser — Monique Teillaud

N° 7004

Juillet 2009

Thème SYM

ISSN 0249-6399 ISRN INRIA/RR--7004--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SOPHIA ANTIPOLIS - MÉDITERRANÉE

Robust and Efficient Delaunay triangulations of points on or close to a sphere

Manuel Caroli* , Pedro M. M. de Castro* , Sébastien Loriot* ,
Camille Wormser† , Monique Teillaud*

Thème SYM — Systèmes symboliques
Équipes-Projets Géométrica

Rapport de recherche n° 7004 — Juillet 2009 — 17 pages

Abstract: We propose two approaches for computing the Delaunay triangulation of points on a sphere, or of *rounded* points close to a sphere, both based on the classic incremental algorithm initially designed for the plane. The space of circles gives the mathematical background for this work. We implemented the two approaches in a fully robust way, building upon existing generic algorithms provided by the CGAL library. The efficiency and scalability of the method is shown by benchmarks.

Key-words: Computational Geometry, Delaunay Triangulation, Voronoi Diagram, Sphere, Space of Circles, Exact Geometric Computing, CGAL

This work was partially supported by the ANR *(Agence Nationale de la Recherche)* under the "Triangles" project of the *Programme blanc* ANR-07-BLAN-0319 `http://www.inria.fr/geometrica/collaborations/triangles/`.

* INRIA Sophia Antipolis – Méditerranée [Email: {Manuel.Caroli, Pedro.Machado, Sebastien.Loriot, Monique.Teillaud}@sophia.inria.fr]
† ETH Zürich, Switzerland [Email: Camille.Wormser@inf.ethz.ch]

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 robuste et efficace, pour des points sur la sphère ou proche d'elle

Résumé : Nous proposons deux façons de calculer la triangulation de Delaunay d'un ensemble de points qui appartiennent soit à la sphère, soit à son voisinage. Ces deux méthodes reposent sur l'algorithme incrémental classique, tel qu'il a été créé à l'origine pour calculer les triangulations de Delaunay planaires. Le cadre mathématique classique justifiant cette approche est rappelé, à l'aide de l'espace des cercles. Ces deux approches ont été implantées de façon robuste en s'appuyant sur les algorithmes génériques fournis par la bibliothèque CGAL. Des tests comparatifs montrent l'efficacité de nos implantations sur des jeux de données de taille variée.

Mots-clés : Géométrie algorithmique, Triangulation de Delaunay, Diagramme de Voronoï, Sphère, Espace des cercles, Calcul géométrique exacte, CGAL

1 Introduction

The CGAL project [cga] provides users with a public discussion mailing list, where they are invited to post questions and express their needs. There are recurring requests for a package computing the Delaunay triangulation or its dual, the Voronoi diagram, of points on a sphere. This is useful for various application domains such as geology, geographic information systems or structural molecular biology, to name a few. An easy and standard answer consists in computing the 3D convex hull of the points, which is notoriously equivalent [Bro80, Sug02]. The convex hull is one of the most popular structures in computational geometry [dBvKOS00, BY98]; optimal algorithms and efficient implementations are available [hul, qhu].

Another fruitful way to compute Delaunay on a sphere consists of reworking known algorithms designed for computing triangulations in $\mathbb{R}^2$. Renka adapts the distance in the plane to a geodesic distance on a sphere and triangulates points on a sphere [Ren97] through the well-known flipping algorithm for Delaunay triangulations in $\mathbb{R}^2$ [Law77]. As a by-product of their algorithm for arrangements of circular arcs, Fogel et al. can compute Voronoi diagrams in the restricted case of points lying exactly on the sphere and having rational coordinates [FSH08, FS, BFH+09b, BFH+09a]. Using two inversions allows Na et al. to reduce the computation of a Voronoi diagram of sites on a sphere to computing two Voronoi diagrams in $\mathbb{R}^2$ [NLC02], but no implementation is available. Note that this method assumes that data points are lying exactly on a sphere.

As we are motivated by applications, we take practical issues into account carefully. While data points lying exactly on the sphere can be provided either by using Cartesian coordinates represented by a number type capable of handling algebraic numbers exactly, or by using spherical coordinates, in practice data-sets in Cartesian coordinates with double precision are most common. In this setting, the data consists of rounded points that do not exactly lie on the sphere, but close to it.

In Section 4, we propose two different approaches to handle such rounded data. Both approaches adapt the well-known incremental algorithm [Bow81] to the case of points on, or close to the sphere. It is important to notice that, even though data points are rounded, we follow the exact geometric computation paradigm pioneered by C. K. Yap [YD95]. Indeed, it is now well understood that simply relying on floating point arithmetic for algorithms of this type is bound to fail (see [KMP+08] for instance).

- The first approach (Section 4.1) considers as input the projections of the rounded data points onto the sphere. Their coordinates are algebraic numbers of degree two. The approach computes the Delaunay triangulation of these points exactly lying on the sphere.
- The second approach (Section 4.2) considers circles on the sphere as input. The radius of a circle (which can alternatively be seen as a *weighted*

point) depends on the distance of the corresponding point to the sphere. The approach computes the weighted Delaunay triangulation of these circles on the sphere, also known as the *regular* triangulation, which is the dual of the Laguerre Voronoi diagram on the sphere [Sug02] and the convex hull of the rounded data points.

These interpretations of rounded data are supported by the space of circles [Ber87, DMT92] (Section 3).

The same algorithm being used for both approaches, only the interpretation of the data and the way in which basic computations on them are performed modify the behavior and possibly the result of the algorithm: Some circles may be hidden in a regular triangulation. We show in Section 4.2 that this can happen only if the rounded points are extremely close to one another. Thus, under some sampling conditions, the two approaches compute the same triangulation (except maybe in degenerate cases for which the considered triangulations are known not to be uniquely defined), and in general, all points appear in the result of both approaches.

In Section 5, we show that the genericity of CGAL [FT06] allows to reuse a large part of the existing package for triangulations in $\mathbb{R}^2$ to implement our two approaches. We present experimental results on very large data-sets, showing the efficiency of our approach. We compare our code to software designed for computing Delaunay triangulations on the sphere, and to convex hull software [HS09, PT09b, hul, sug, qhu, Ren97, FS]. The performance, robustness, and scalability of our approaches show their added value.

2 Definitions and Notation

Let us first recall the definition of the *regular triangulation* in $\mathbb{R}^2$, also known as *weighted Delaunay triangulation*. A circle c with center $p \in \mathbb{R}^2$ and squared radius r^2 is considered equivalently as a *weighted point* and is denoted by $c = (p, r^2)$. The *power product* of $c = (p, r^2)$ and $c' = (p', r'^2)$ is defined as $pow(c, c') = \|pp'\|^2 - r^2 - r'^2$, where $\|pp'\|$ denotes the Euclidean distance between p and p'. Circles c and c' are orthogonal iff $pow(c, c') = 0$. If $pow(c, c') > 0$ (i.e. the disks defined by c and c' do not intersect, or the circles intersect with an angle strictly smaller than $\frac{\pi}{2}$), we say that c and c' are *suborthogonal*. If $pow(c, c') < 0$, then we say that c and c' are *superorthogonal* (see Figure 1). Three circles whose centers are not collinear have a unique common orthogonal circle.

Let $\mathcal{S}$ be a set of circles. Given three circles of $\mathcal{S}$, $c_i = (p_i, r_i^2), i \in I \subset \mathbb{N}$ and $|I| = 3$, whose centers are not collinear, let T_I be the triangle whose vertices are the three centers $p_i, i \in I$. We define the *orthogonal circle* of T_I as the circle that is orthogonal to the three circles $c_i, i \in I$. T_I is said to be *regular* if for any circle $c \in \mathcal{S}$, the orthogonal circle of T_I and c are not superorthogonal. A *regular triangulation*

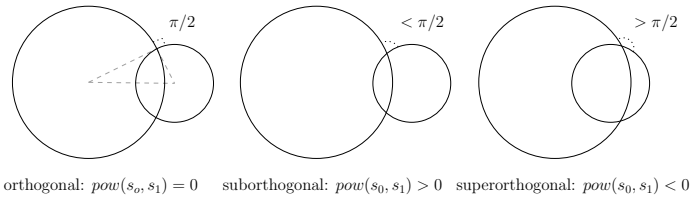

Figure 1: Orthogonal, suborthogonal and superorthogonal circles in $\mathbb{R}^2$.

$\mathcal{RT}(\mathcal{S})$ is a partition of the convex hull of the centers of the circles of $\mathcal{S}$ into regular triangles formed by these centers. See Figure 2 for an example. The dual of the regular triangulation is known as the *power diagram*, *weighted Voronoi diagram*, or *Laguerre diagram*.

If all radii are equal, then the power test reduces to testing whether a point lies inside, outside, or on the circle passing through three points; the regular triangulation of the circles is the Delaunay triangulation $\mathcal{DT}$ of their centers.

More background can be found in [Aur87]. We refer the reader to standard textbooks for algorithms computing Delaunay and regular triangulations [dBvKOS00, BY98].

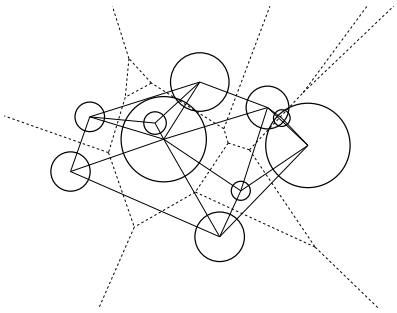

Figure 2: Regular triangulation of a set of circles in the plane (their power diagram is shown dashed)

This definition generalizes in a natural manner to the case of circles lying on a sphere $\mathbb{S}$ in $\mathbb{R}^3$: Angles between circles are measured on the sphere, triangles are drawn on the sphere, their edges being arcs of great circles. As can be seen in the next section, the space of circles provides a geometric presentation showing without any computation that the regular triangulation on $\mathbb{S}$ is a convex hull in $\mathbb{R}^3$ [Sug02].

In the sequel, we assume that $\mathbb{S}$ is given by its center, having rational coordinates (we take the origin O without loss of generality), and a rational squared radius R^2. This is also how spheres are represented in CGAL.[1]

3 Space of Circles

Computational geometers are familiar with the classic idea of lifting up sites from the Euclidean plane onto the unit paraboloid Π in $\mathbb{R}^3$ [Aur91]. We quickly recall the notion of *space of circles* here and refer to the literature for a more detailed presentation [DMT92]. In this lifting, points of $\mathbb{R}^3$ are viewed as circles of $\mathbb{R}^2$ in the space of circles: A circle $c = (p, r^2)$ in $\mathbb{R}^2$ is

[1] We mention rational numbers to simplify the presentation. CGAL allows more general number types that provide field operations: $+, -, \times, /$.

mapped by π to the point $\pi(c) = (x_p, y_p, x_p^2 + y_p^2 - r^2) \in \mathbb{R}^3$. A point of $\mathbb{R}^3$ lying respectively outside, inside, or on the paraboloid Π represents a circle with respectively positive, imaginary, or null radius. The circle c in $\mathbb{R}^2$ corresponding to a point $\pi(c)$ of $\mathbb{R}^3$ outside Π is obtained as the projection onto $\mathbb{R}^2$ of the intersection between Π and the cone formed by lines through $\pi(c)$ that are tangent to Π; this intersection is also the intersection of the polar plane $P(c)$ of $\pi(c)$ with respect to the quadric Π.

Points lying respectively on, above, below $P(c)$ correspond to circles in $\mathbb{R}^2$ that are respectively orthogonal, suborthogonal, superorthogonal to c. The predicate $pow(c, c')$ introduced above is thus equivalent to the orientation predicate in $\mathbb{R}^3$ that tests whether the point $\pi(c')$ lies on, above or below the plane $P(c)$. If c is the common orthogonal circle to three input circles c_1, c_2, and c_3 (where $c_i = (p_i, r_i^2)$ for each i), then $pow(c, c')$ is the orientation predicate of the four points $\pi(c_1), \pi(c_2), \pi(c_3), \pi(c')$ of $\mathbb{R}^3$. It can be expressed as

$$\text{sign} \begin{vmatrix} 1 & 1 & 1 & 1 \\ x_{p_1} & x_{p_2} & x_{p_3} & x_{p'} \\ y_{p_1} & y_{p_2} & y_{p_3} & y_{p'} \\ z_{p_1} & z_{p_2} & z_{p_3} & z_{p'} \end{vmatrix}, \tag{1}$$

where $z_{p_i} = x_{p_i}^2 + y_{p_i}^2 - r_i^2$ for each i and $z_{p'}^2 = x_{p'}^2 + y_{p'}^2 - r'^2$. It allows to relate Delaunay or regular triangulations in $\mathbb{R}^2$ and convex hulls in $\mathbb{R}^3$ [Aur91], while Voronoi diagrams in $\mathbb{R}^2$ are related to upper envelopes of planes in $\mathbb{R}^3$.

Up to a projective transformation, a sphere in $\mathbb{R}^3$ can be used for the lifting instead of the usual paraboloid [Ber87]. In this representation, common in geometry, the sphere has a pole[2] and can be identified to the Euclidean plane $\mathbb{R}^2$. What we are interested in this paper is the space of circles drawn on the sphere $\mathbb{S}$ itself, without any pole. This space of circles has a nice relation to the de Sitter space in Minkowskian geometry [Cox43]. We can still construct the circle c on $\mathbb{S}$ that is associated to a point $p = \pi_{\mathbb{S}}(c)$ of $\mathbb{R}^3$ as the intersection between $\mathbb{S}$ and the polar plane $P_{\mathbb{S}}(p)$ of p with respect to the quadric $\mathbb{S}$ (Figure 3). Its center is the projection of p onto $\mathbb{S}$ and its squared radius is $\|Op\|^2 - R^2$ (as above, imaginary radii are possible).[3] So, in the determinant in (1), x_{p_i}, y_{p_i}, and z_{p_i} (respectively $x_{p'}, y_{p'}, z_{p'}$) are precisely the coordinates of the points $p_i = \pi_{\mathbb{S}}(c_i)$ (respectively $p' = \pi_{\mathbb{S}}(p)$). This will be extensively used in Section 4. Again, we remark that Delaunay and regular triangulations on $\mathbb{S}$ relate to convex hulls in 3D.

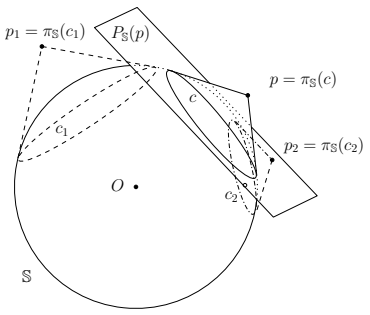

Figure 3: c_1 is suborthogonal to c, c_2 is superorthogonal to c.

[2] See the nice treatment of infinity in [Ber87].

[3] Remember that $\mathbb{S}$ is centered at O and has squared radius R^2.

Interestingly, rather than using a convex hull algorithm to obtain the Delaunay or regular triangulation on the surface as usually done for $\mathbb{R}^2$ [Aur91], we will do the converse in the next section.

4 Algorithm

The incremental algorithm for computing a regular triangulation of circles on the sphere $\mathbb{S}$ is a direct adaptation of the algorithm in $\mathbb{R}^2$ [Bow81]. Assume that $\mathcal{RT}_{i-1} = \mathcal{RT}(\{c_j \in \mathcal{S}, j = 1, \ldots, i-1\})$ has been computed.[4] The insertion of $c_i = (p_i, r_i^2)$ works as follows:
• locate p_i (i.e. find the triangle t containing p_i),
• `if` t is hiding p_i (i.e. if c_i and the orthogonal circle of t are suborthogonal) then `stop`; p_i is not a vertex of $\mathcal{RT}_i$. Note that this case never occurs for Delaunay triangulations.
• `else` (i) find all triangles whose orthogonal circles are superorthogonal to c_i and remove them; this forms a polygonal region that is star-shaped with respect to p_i;[5] (ii) triangulate the polygonal region just created by constructing the triangles formed by the boundary edges of the region and the point p_i.
Steps (i) and (ii) can alternatively be accomplished by flipping [ES96], which is the version currently implemented in the CGAL package [Yvi09].

Two main predicates are used by this algorithm:
The *orientation* predicate allows to check the orientation of three points p, q, r on the sphere (this predicate is used in particular to locate new points). It is equivalent to computing the side of the plane defined by O, p, q on which r is lying, i.e. the orientation of O, p, q, r in $\mathbb{R}^3$.
The *power test* introduced in Section 2 (this predicate is used to identify the triangles whose orthogonal circles are superorthogonal to each new circle) boils down to an orientation predicate in $\mathbb{R}^3$, as seen in Section 3.

The two approaches quickly presented in the introduction fall into the general framework of computing the regular triangulation of circles on the sphere. The next two sections show more precisely how these predicates are evaluated in each approach.

4.1 First approach: Using points on the sphere

In this approach, input points for the computation are chosen to be the projections on $\mathbb{S}$ of the rounded points of the data-set with rational coordinates.

[4] For the sake of simplicity, we assume that the center O of $\mathbb{S}$ lies in the convex hull of the data-set. This is likely to be the case in practical applications. So, we just initialize the triangulation with four dummy points that contain O in their convex hull and can optionally be removed in the end.

[5] As previously noted for the edges of triangles, all usual terms referring to segments are transposed to arcs of great circles on the sphere.

The three coordinates of an input point are thus algebraic numbers of degree two lying in the same extension field of the rationals.

In this approach weights, or equivalently radii if circles, are null. The power test consists in this case in answering whether a point s lies inside, outside,[6] or on the circle passing through p, q, r on the sphere. Following Section 3, this is given by the orientation of p, q, r and s since points on the sphere are mapped to themselves by $\pi_{\mathbb{S}}$.

The difficulty comes from the fact that input points have algebraic coordinates. The coordinates of two different input points on the sphere are in general lying in different extensions. Then the 3D orientation predicate of p, q, r, s given by (1) is the sign of an expression lying in an algebraic extension of degree 16 over the rationals, of the form $a_1\sqrt{\alpha_1} + a_2\sqrt{\alpha_2} + a_3\sqrt{\alpha_3} + a_4\sqrt{\alpha_4}$ where all a's and α's are rational. Evaluating this sign in an exact way allows to follow the exact computation framework ensuring the robustness of the algorithm.

Though software packages offer exact operations on general algebraic numbers [cor, led], they are much slower than computing with rational numbers. The sign of the above simple expression can be computed as follows:
–1– evaluate the signs of $A_1 = a_1\sqrt{\alpha_1}+a_2\sqrt{\alpha_2}$ and $A_2 = a_3\sqrt{\alpha_3}+a_4\sqrt{\alpha_4}$, by comparing $a_i\sqrt{\alpha_i}$ with $a_{i+1}\sqrt{\alpha_{i+1}}$ for $i = 1, 3$, which reduces after squaring to comparing two rational numbers,
–2– the result follows if A_1 and A_2 have different signs,
–3– otherwise, compare A_1^2 with A_2^2, which is an easier instance of –1–.
To summarize, the predicate is given by the sign of polynomial expressions in the rational coordinates of the rounded data points, which can be computed exactly using rational numbers only.

4.2 Second approach: Using weighted points

In this approach, the regular triangulation of the weighted points is computed as described above. As in the previous approach, both predicates (orientation on the sphere and power test) reduce to orientation predicates on the data points in $\mathbb{R}^3$. Note that Section 3 showed that the weight $\|Op\|^2 - R^2$ of a point p stays implicit, it need not be explicitly computed.

Depending on the weights, some points can be hidden in a regular triangulation. We prove now that under some sampling conditions on the rounded data, there is actually no hidden point.

Lemma 4.1. *Let us assume that all data points lie within a distance δ from $\mathbb{S}$. If the distance between any two points is larger than $2\sqrt{R\delta}$, then no point is hidden.*

[6] On $\mathbb{S}$, the interior (respectively exterior) of a circle c that is not a great circle of $\mathbb{S}$ corresponds to the interior (respectively exterior) of the half-cone in 3D whose apex is the center of $\mathbb{S}$ and that intersects $\mathbb{S}$ along c.

Proof. A point is hidden iff it is contained inside the 3D convex hull of the set of data points $\mathcal{S}$. Let p be a data point, at distance ρ from O. We have $\rho \in [R-\delta, R+\delta]$. Denote by d_p the minimum distance between p and the other points. If $d_p > \sqrt{(R+\delta)^2 - \rho^2}$, the set $B(O, R+\delta) \setminus B(p, d_p)$ is included in the half-space $H^+ = \{q : \langle q-p, O-p\rangle > 0\}$. Under these conditions, all other points belong to H^+ and p is not inside the convex hull of the other points. It follows that if the distance between any two data points is larger than $\sup_\rho \sqrt{(R+\delta)^2 - \rho^2} = 2\sqrt{R\delta}$, no point is hidden. □

Let us now assume we use double precision floating point numbers as defined in the IEEE standard 754 [iee08, Gol91]. The mantissa is encoded using 52 bits. Let γ denote the worst error, for each Cartesian coordinate, done while rounding a point on $\mathbb{S}$ to the nearest point whose coordinates can be represented by double precision floating point numbers. Let us use the standard term $\mathrm{ulp}(x)$ denoting the gap between the two floating-point numbers closest to the real value x [Mul05]. Assuming again that the center of $\mathbb{S}$ is O, one has $\gamma \leq \mathrm{ulp}(R) = 2^{-52+\lfloor \log_2(R) \rfloor} \leq 2^{-52}R$. Then, δ in the previous lemma is such that $\delta \leq \sqrt{3/4}\gamma < 2^{-52}R$. Using the result of the lemma, no point is hidden in the regular triangulation as soon as the distance between any two points is greater than $2^{-25}R$, which is highly probable in practice.

5 Implementation and Experiments

The two methods presented in Section 4 have been implemented in C++, based on the CGAL package that computes triangulations in $\mathbb{R}^2$. The package introduces an infinite vertex in the triangulation to compactify $\mathbb{R}^2$. Thus the underlying combinatorial triangulation is a triangulation of the topological sphere. This allows us to reuse the whole combinatorial part of CGAL 2D triangulations [PY09] without any modification. However, the geometric embedding itself [Yvi09], bound to $\mathbb{R}^2$, must be modified in an easy but non-negligible way, by removing any reference to the infinite vertex. A similar work was done to compute triangulations in the 3D flat torus [CT09b, CT09a], reusing the CGAL 3D triangulation package [PT09a, PT09b] as much as possible.

Also, the genericity offered in CGAL by the mechanism of traits classes, that encapsulate the geometric predicates needed by the algorithms, allows us to easily use exactly the same algorithm with two different traits classes for our two approaches.

To display the triangulation and its dual, the code is interfaced with the CGAL 3D spherical kernel [dCT09, dCCLT09], which provides primitives on circular arcs in 3D. The vertices of the triangulations shown are the projections on the sphere of the rounded data points. The circular arcs are drawn on the surface of the sphere (see Figures 4, 6 and 7).

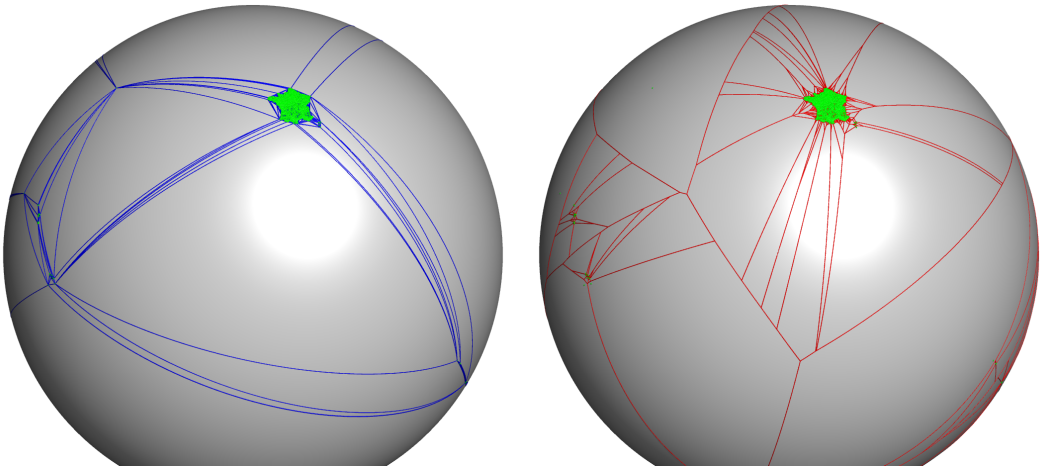

Figure 4: Delaunay triangulation (left) and Voronoi diagram (right) of the 9031 post-offices in France (including Overseas Departments and Territories).

We compare the running time of our methods with several available software packages[7]. On Figure 5, we consider large sets of random data points[8] (up to 2^{23}) on the sphere, rounded to double coordinates. Figure 6 quickly mentions running times on some real-life data.

Graph 1st of Figure 5 shows the results of our first approach. We coded a traits class implementing the exact predicates presented in Section 4.1, together with semi-static and dynamic filtering [LPY05]. The non-linearity of the behavior is due to the fact that semi-static filters hardly ever fail for less than 2^{13} points, and almost always fail for more than 2^{18} points.

Graph 2nd shows the results of the second approach. One of the predefined kernels[9] of CGAL provides us directly with an exact implementation of the predicates, filtered both semi-statically and dynamically. We observe that no point is hidden with such distributions, even when the data-set is large, which confirms in practice the discussion of Section 4.2.

The CGAL 3D Delaunay triangulation (graph DT3) [PT09b] also provides this convex hull as a by-product. We use the same CGAL kernel and insert the center of the sphere to avoid penalizing this code with too many degenerate cases that would always cause filters to fail.
For these three approaches, 3D spatial sorting reduces the running time of the location step of point insertion [Del09, Buc09].

[7] Times reported are in seconds. Of course, the time of input/ouput is not counted. Computations on an Mac Book Pro 3,1 MAC OS X version 10.5.7, Processor 2.6 GHz Intel Core 2 Duo, Memory 2 Go 667 MHz DDR2 SDRAM. Compiler `g++ 4.3.2` with -O3 and -DNDEBUG flags. Compiler `g77 3.4.3` with -O3 for Fortran.

[8] generated by `CGAL::Random_points_on_sphere_3`

[9] precisely `CGAL::Exact_predicates_inexact_constructions_kernel`

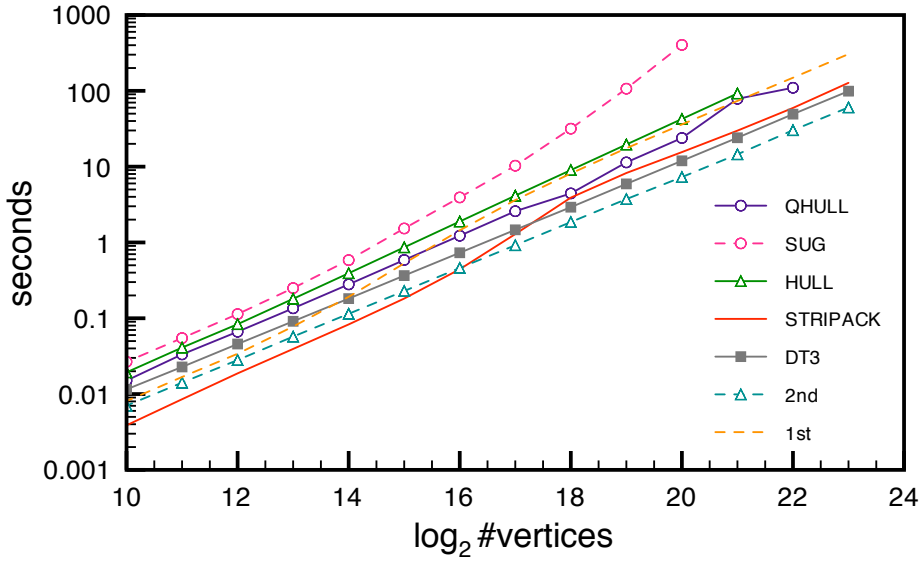

Figure 5: Comparative benchmarks. The programs were aborted when their running time was above 10 minutes.

No graph is shown for the code of Fogel and Setter [FS, FSH08]: It is preliminary and has not been fully-optimized yet; its running time is close to 600 seconds for 2^{12} points. Note that it is restricted to points with rational coordinates lying precisely on a sphere.

We consider the following two software packages computing a convex hull in 3D,[10] for which the data points are first rounded to points with integer coordinates. Predicates are evaluated exactly using single precision computations.

Graph HULL corresponds to the code [hul] of Clarkson, who uses a randomized incremental construction [CMS93] with an exact arithmetic on integers [Cla92].

Graph SUG shows the running times of Sugihara's code in Fortran [sug, Sug02].

Graph QHULL shows the performance of the famous Qhull package of Barber et al. [qhu] when computing the 3D convex hull of the points. The option we use handles round-off errors from floating point arithmetic by merging facets of the convex hull when necessary.

Renka computes the triangulation with an algorithm similar to our first approach, but his software STRIPACK, in Fortran, uses approximate computations in double [Ren97]. Consequently, it performs quite well on random

[10] The plot does not show the results of the CGAL 3D convex hull package [HS09], whose implementation dates back to several years, because it was much slower than all other methods (roughly 500 times slower than QHULL).

points (better than our implementations for small random data-sets), but it fails on some data-sets like the sets shown in Figures 6 and 7. Our implementations handle even highly degenerate cases.

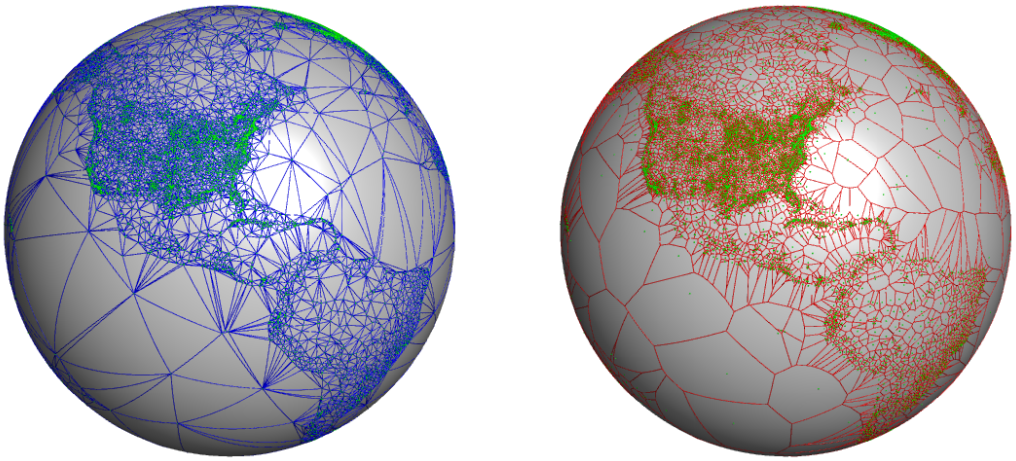

Figure 6: Delaunay triangulation (top) and Voronoi diagram (bottom) of 20950 weather stations all around the world. Data and more information can be found at `http://www.locationidentifiers.org/`. Our second approach computes the result in 0.14 seconds, while Qhull needs 0.35 seconds, and the first approach 0.57 seconds. STRIPACK fails on this data-set.

To test for exactness we devised a point set that is especially hard to triangulate because it yields many sliver-shaped triangles in the triangulation. This point set is defined as

$$\begin{aligned}\mathcal{S}_n \;=\; & \left\{ \begin{pmatrix} \cos\theta\sin\phi \\ \sin\theta\sin\phi \\ \cos\phi \end{pmatrix} \;\middle|\; \theta \in \left\{0, \frac{\pi}{n}, \ldots, \frac{(n-1)\pi}{n}, \pi\right\}, \phi = \frac{(\theta^2+1)}{\pi^2} \right\} \\ & \cup \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, -\frac{1}{\sqrt{3}} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right\}\end{aligned}$$

In Figure 7 we show the Delaunay triangulation of $\mathcal{S}_{250}$ and the Voronoi diagram of $\mathcal{S}_{100}$.

Conclusion

The results show that our second approach yields better timings than all the other tested software packages for large data-sets, while being fully robust. Note that it can in fact be used as well to compute the convex hull of points that are not close to a sphere: The center of the sphere can be

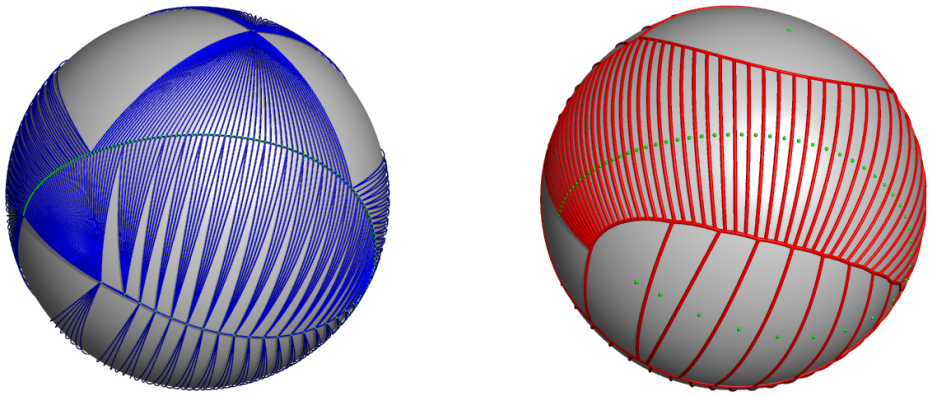

Figure 7: Delaunay triangulation of $\mathcal{S}_{250}$ (left), Voronoi diagram of $\mathcal{S}_{100}$ (right). STRIPACK fails for e.g. $n = 1,500$.

chosen at any point inside a tetrahedron formed by any four non-coplanar data points.

The first approach is slower but still one of the most scalable. It is the only one that exactly computes the triangulation on the sphere for input points with algebraic coordinates, and thus ensures that in any case all points will appear in the triangulation.

Acknowledgments

We warmly acknowledge Ophir Setter and Efi Fogel who kindly worked on their code to give us access to it. We wish to thank Jean-Marc Schlenker for very interesting discussions on the de Sitter space.

References

[Aur87] Franz Aurenhammer. Power diagrams: properties, algorithms and applications. *SIAM Journal of Computing*, 16:78–96, 1987.

[Aur91] Franz Aurenhammer. Voronoi diagrams: A survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405, September 1991.

[Ber87] Marcel Berger. The space of spheres. In *Geometry (vols. 1-2)*, pages 349–361. Springer-Verlag, 1987.

[BFH+09a] Eric Berberich, Efi Fogel, Dan Halperin, Michael Kerber, and Ophir Setter. Arrangements on parametric surfaces II: Concretizations and applications, 2009. Submitted.

[BFH+09b] Eric Berberich, Efi Fogel, Dan Halperin, Kurt Mehlhorn, and Ron Wein. Arrangements on parametric surfaces I: General framework and infrastructure, 2009. Submitted.

[Bow81] A. Bowyer. Computing Dirichlet tessellations. *The Computer Journal*, 24(2):162–166, 1981.

[Bro80] K. Q. Brown. *Geometric transforms for fast geometric algorithms*. Ph.D. thesis, Dept. Comput. Sci., Carnegie-Mellon Univ., Pittsburgh, PA, 1980. Report CMU-CS-80-101.

[Buc09] Kevin Buchin. Constructing Delaunay triangulations along space-filling curves. In *Proceedings European Symposium on Algorithms*, 2009. To appear.

[BY98] Jean-Daniel Boissonnat and Mariette Yvinec. *Algorithmic Geometry*. Cambridge University Press, UK, 1998. Translated by Hervé Brönnimann.

[cga] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[Cla92] K. L. Clarkson. Safe and effective determinant evaluation. In *Proceedings 33rd Annual IEEE Symposium on Foundations of Computer Science*, pages 387–395, October 1992.

[CMS93] K. L. Clarkson, K. Mehlhorn, and R. Seidel. Four results on randomized incremental constructions. *Computational Geometry: Theory and Applications*, 3(4):185–212, 1993.

[cor] CORE number library. http://cs.nyu.edu/exact/core_pages.

[Cox43] H. S. M. Coxeter. A geometrical background for de Sitter's world. *American Mathematical Monthly*, 50:217–228, 1943.

[CT09a] Manuel Caroli and Monique Teillaud. 3D periodic triangulations. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 3.5 edition, 2009. To appear.

[CT09b] Manuel Caroli and Monique Teillaud. Computing 3D periodic triangulations. In *Proceedings 17th European Symposium on Algorithms*, 2009. To appear. Full version available as INRIA Reserch Report No 6823, http://hal.inria.fr/inria-00356871.

[dBvKOS00] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, Germany, 2nd edition, 2000.

[dCCLT09] Pedro M. M. de Castro, Frédéric Cazals, Sébastien Loriot, and Monique Teillaud. Design of the CGAL 3D Spherical Kernel and application to arrangements of circles on a sphere. *Computational Geometry : Theory and Applications*, 42(6-7):536–550, 2009.

[dCT09] Pedro M. M. de Castro and Monique Teillaud. 3D spherical geometry kernel. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 3.4 edition, 2009.

[Del09] Christophe Delage. Spatial sorting. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 3.4 edition, 2009.

[DMT92] Olivier Devillers, Stefan Meiser, and Monique Teillaud. The space of spheres, a geometric tool to unify duality results on Voronoi diagrams. In *Proceedings 4th Canadian Conference on Computational Geometry*, pages 263–268, 1992. Full version available as INRIA Research Report No 1620, http://hal.inria.fr/inria-00074941.

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

[FS] Efi Fogel and Ophir Setter. Software for Voronoi diagram on a sphere. Personal communication.

[FSH08] Efi Fogel, Ophir Setter, and Dan Halperin. Exact implementation of arrangements of geodesic arcs on the sphere with applications. In *Abstracts of 24th European Workshop on Computational Geometry*, pages 83–86, 2008.

[FT06] Efi Fogel and Monique Teillaud. Generic programming and the CGAL library. In Jean-Daniel Boissonnat and Monique Teillaud, editors, *Effective Computational Geometry for Curves and Surfaces*. Springer-Verlag, Mathematics and Visualization, 2006.

[Gol91] D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48, March 1991.

[HS09] Susan Hert and Stefan Schirra. 3D convex hulls. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 3.4 edition, 2009.

[hul] Hull, a program for convex hulls. http://www.netlib.org/voronoi/hull.html.

[iee08] IEEE standard for floating-point arithmetic. *IEEE Std 754-2008*, pages 1–58, August 2008.

[KMP+08] Lutz Kettner, Kurt Mehlhorn, Sylvain Pion, Stefan Schirra, and Chee Yap. Classroom examples of robustness problems in geometric computations. *Computational Geometry: Theory and Applications*, 40:61–78, 2008.

[Law77] 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.

[led] LEDA, Library for efficient data types and algorithms. http://www.algorithmic-solutions.com/enleda.htm.

[LPY05] C. Li, S. Pion, and C. K. Yap. Recent progress in exact geometric computation. *Journal of Logic and Algebraic Programming*, 64(1):85–111, 2005.

[Mul05] Jean-Michel Muller. On the definition of ulp(x). Research Report 5504, INRIA, February 2005. http://hal.inria.fr/inria-00070503/.

[NLC02] Hyeon-Suk Na, Chung-Nim Lee, and Otfried Cheong. Voronoi diagrams on the sphere. *Computational Geometry: Theory and Applications*, 23:183–194, 2002.

[PT09a] Sylvain Pion and Monique Teillaud. 3D triangulation data structure. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 3.4 edition, 2009.

[PT09b] Sylvain Pion and Monique Teillaud. 3D triangulations. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 3.4 edition, 2009.

[PY09] Sylvain Pion and Mariette Yvinec. 2D triangulation data structure. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 3.4 edition, 2009.

[qhu] Qhull. http://www.qhull.org/.

[Ren97] Robert J. Renka. Algorithm 772: STRIPACK: Delaunay triangulation and Voronoi diagram on the surface of a sphere. *ACM Transactions on Mathematical Software*, 23(3):416–434, 1997. Software available at http://orion.math.iastate.edu/burkardt/f_src/stripack/stripack.html.

[sug] Three-dimensional convex hulls. http://www.simplex.t.u-tokyo.ac.jp/~sugihara/opensoft/opensofte.html.

[Sug02] Kokichi Sugihara. Laguerre Voronoi diagram on the sphere. *Journal for Geometry and Graphics*, 6(1):69–81, 2002.

[YD95] 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.

[Yvi09] Mariette Yvinec. 2D triangulations. In CGAL Editorial Board, editor, *CGAL User and Reference Manual*. 3.4 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