Robust and Efficient Delaunay Triangulations of Points on Or Close to a Sphere

Manuel Caroli, Pedro M M de Castro, Sébastien Loriot, Olivier Rouiller, Monique Teillaud, Camille Wormser

Robust and Efficient Delaunay Triangulations of Points on Or Close to a Sphere*

Manuel Caroli[1], Pedro M.M. de Castro[1], Sébastien Loriot[1], Olivier Rouiller[1], Monique Teillaud[1], and Camille Wormser[2]

[1] INRIA Sophia Antipolis – Méditerranée, France
{Manuel.Caroli,Pedro.Machado,Monique.Teillaud}@sophia.inria.fr
[2] ETH Zürich, Switzerland
Camille.Wormser@inf.ethz.ch

Abstract. We propose two ways to compute 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. We use the so-called space of circles as mathematical background for this work. We present a fully robust implementation built upon existing generic algorithms provided by the CGAL library. The efficiency of the implementation is established by benchmarks.

1 Introduction

The CGAL project [3] 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 of points on a sphere or its dual, the Voronoi diagram. This is useful in many domains such as geology, geographic information systems, information visualization, or structural molecular biology, to name a few. An easy and standard solution to the problem of computing such a Delaunay triangulation consists in constructing the 3D convex hull of the points: They are equivalent [13,38]. The convex hull is one of the most popular structures in computational geometry [20,11]; optimal algorithms and efficient implementations are available [1,2].

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 [37] through the well-known flipping algorithm for Delaunay triangulations in $\mathbb{R}^2$ [30]. As a by-product of their algorithm for arrangements of circular arcs, Fogel et al. can compute Voronoi diagrams of points lying exactly on the sphere [26]. 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

* 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/

in $\mathbb{R}^2$ [33], 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 ways to handle such rounded data. Both approaches adapt the well-known incremental algorithm [12] 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 [39]. Indeed, it is now well understood that simply relying on floating point arithmetic for algorithms of this type is bound to fail (see [29] 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 [38] and the convex hull of the rounded-data points.

These interpretations of rounded data presented in this work are supported by the space of circles [10,24] (Section 3).

We implemented both approaches, taking advantage of the genericity of CGAL. In Section 5, we present experimental results on very large data-sets, showing the efficiency of our approaches. We compare our code to software designed for computing Delaunay triangulations on the sphere, and to convex-hull software [28,35,1,6,2,37,25]. The performance, robustness, and scalability of our approaches express 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

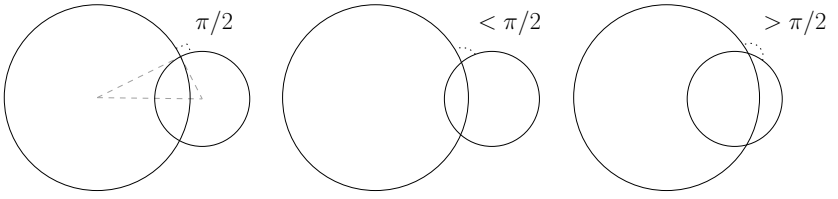

Fig. 1. From left to right: orthogonal ($pow(s_0, s_1) = 0$), suborthogonal ($pow(s_0, s_1) > 0$), and superorthogonal ($pow(s_0, s_1) < 0$) circles in $\mathbb{R}^2$

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 $\mathcal{S}$, $c_i = (p_i, r_i^2)$, $i = 1 \ldots 3$, whose centers are not collinear, let T be the triangle whose vertices are the three centers p_1, p_2, and p_3. We define the *orthogonal circle* of T as the circle that is orthogonal to the three circles c_1, c_2, and c_3. T is said to be *regular* if for any circle $c \in \mathcal{S}$, the orthogonal circle of T and c are not superorthogonal. A *regular triangulation* $\mathcal{RT}(\mathcal{S})$ 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 [8]. We refer the reader to standard textbooks for algorithms computing Delaunay and regular triangulations [20,11].

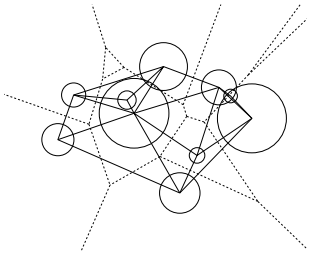

Fig. 2. Regular triangulation of a set of circles in the plane (their power diagram is shown dashed)

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 $\mathbb{R}^3$ [38].

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]

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

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 $\mathbb{R}^3$ [9]. We quickly recall the notion of *space of circles* here and refer to the literature for a more detailed presentation [24]. 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 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 $\mathbb{R}^3$ [9], 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 [10]. In this representation 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 [19].

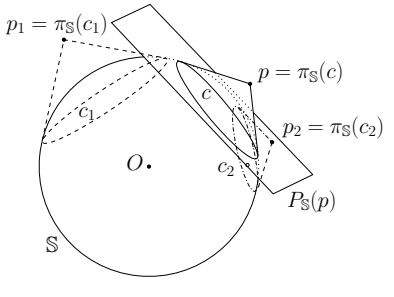

Fig. 3. c_1 is suborthogonal to c, c_2 is superorthogonal to c

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 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

[2] See the nice treatment of infinity in [10].

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

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.

Interestingly, rather than using a convex hull algorithm to obtain the Delaunay or regular triangulation on the surface as usually done for $\mathbb{R}^2$ [9], 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 $\mathbb{R}^2$ [12]. 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.

Two main predicates are used by this algorithm:

The *orientation* predicate allows to check the orientation of three points p, q, and 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, and q on which r is lying, i.e., the orientation of O, p, q, and r in $\mathbb{R}^3$.
The *power test* introduced in Section 2 boils down to an orientation predicate in $\mathbb{R}^3$, as seen in Section 3. (This predicate is used to identify the triangles whose orthogonal circles are superorthogonal to each new circle.)

The two approaches briefly presented in the introduction fall into the general framework of computing the regular triangulation of circles on the sphere. The next two sections precisely show 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. The

[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.

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, and 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, and 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 [4,5], 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 shows that the weight of a point p is implicit, as it does not need to be explicitly computed throughout the entire algorithm.

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 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.*

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

[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.

$\rho \in [R-\delta, R+\delta]$. 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 [7,27]. 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 [32]. 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.

Note that this approach can 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 chosen at any point inside a tetrahedron formed by any four non-coplanar data points.

5 Implementation and Experiments

Both approaches 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 [36] without any modification. However, the geometric embedding itself [40], bound to $\mathbb{R}^2$, must be modified by removing any reference to the infinite vertex. A similar work was done to compute triangulations in the 3D flat torus [16,15], reusing the CGAL 3D triangulation package [34,35] 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 [21,22], 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 5 and 6).

We compare the running time of our approaches with several available software packages on a MacBook Pro 3,1 equipped with a 2.6 GHz Intel Core 2

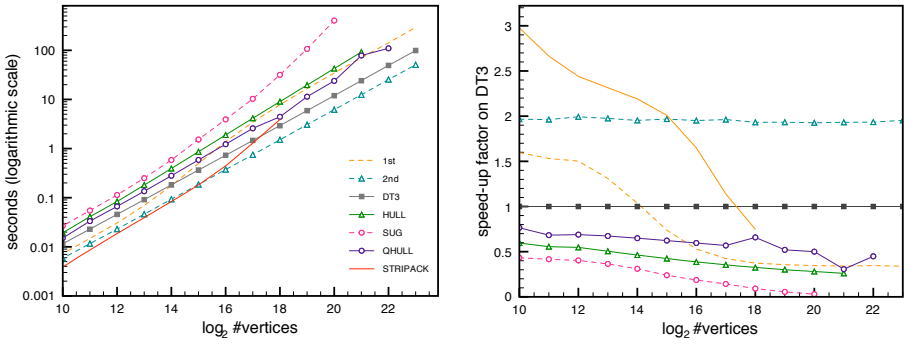

Fig. 4. Comparative benchmarks. The programs were aborted when their running time was above 10 minutes (HULL, SUG, QHULL) or in case of failure (STRIPACK)

processor and 2GB 667 MHz DDR2 SDRAM[7] (see Figure 4). We consider large sets of random data points[8] (up to 2^{23} points) on the sphere, rounded to double coordinates. Figure 5 indicates running times on some real-life data.

Graph 1st of Figure 4 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 [31]. The non-linear behavior of the running time is due to the fact that our 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. In our experiments we have observed 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) [35], with the same CGAL kernel, also provides this convex hull as a by-product. We insert the center of the sphere to avoid penalizing this code with too many predicate calls on five cospherical points 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 [23,14].

If the data points are lying exactly on a sphere, their Delaunay Triangulation can be extracted from an arrangement of geodesic arcs as computed by the code of Fogel and Setter [25,26]. Since it is not the main purpose of their algorithm, the running times are not comparable: close to 600 seconds for 2^{12} points. Note however that the code is preliminary and has not been fully optimized yet. No graph is shown.

[7] Further details: MAC OS X version 10.5.7, 64 bits; compiler `g++ 4.3.2` with -O3 and -DNDEBUG flags, `g77 3.4.3` with -O3 for Fortran. All running times mentioned exclude time used by input/output.

[8] Generated by `CGAL::Random_points_on_sphere_3`

[9] Precisely `CGAL::Exact_predicates_inexact_constructions_kernel`

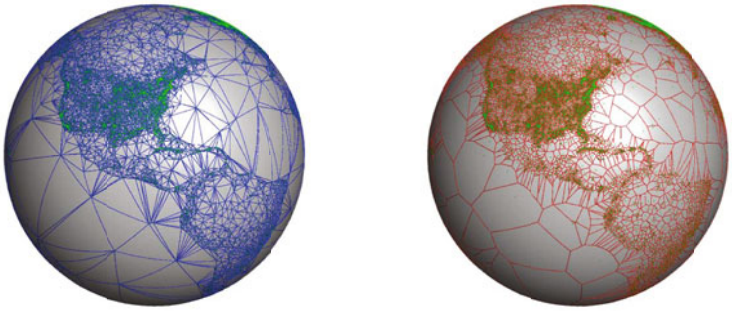

Fig. 5. Delaunay triangulation (left) and Voronoi diagram (right) of 20,950 weather stations all around the world. Data and more information can be found at `http://www.locationidentifiers.org/`. Our second approach computes the result in 0.14 seconds, while Qhull needs 0.35 seconds, and the first approach 0.57 seconds. STRIPACK fails on this data-set.

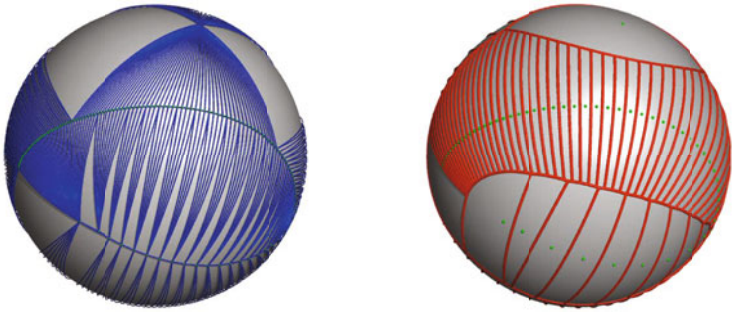

Fig. 6. Delaunay triangulation of $\mathcal{S}_{250}$ (left), Voronoi diagram of $\mathcal{S}_{100}$ (right). STRIPACK fails for e.g. $n = 1,500$.

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 [1] of Clarkson, who uses a randomized incremental construction [18] with an exact arithmetic on integers [17].

Graph SUG shows the running times of Sugihara's code in Fortran [6,38].

Graph QHULL shows the performance of the famous Qhull package of Barber et al. [2] 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. The convex hull of points situated close to the sphere contains in practice all the input points (see Lemma 1). In this situation QHULL is clearly outperformed by the second approach. However, QHULL can

[10] The plot does not show the results of the CGAL 3D convex hull package [28] because it is much slower than all other methods (roughly 500 times slower than QHULL).

be about twice faster than our second approach when almost all the input points are hidden.

Renka computes the triangulation with an algorithm similar to our first approach, but his software STRIPACK, in Fortran, uses approximate computations in double [37]. Consequently, it performs quite well on random points (better than our implementations for small random data-sets), but it fails on some data-sets: Using STRIPACK, we did not manage to compute a triangulation of more than 2^{19} random points (it returns an error flag). The same occurred for the inputs used to produce Figures 5 and 6. Our implementations handle arbitrary data sets.

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

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

In Figure 6 we show the Delaunay triangulation of $\mathcal{S}_{250}$ and the Voronoi diagram of $\mathcal{S}_{100}$.

In Table 1, we compare the memory usage of our two approaches, the 3D Delaunay triangulation, and Qhull.[11] The given figures given in bytes per processed vertex (bppv) and averaged over several data-sets of size larger than 2^{16}.

Table 1. Memory usage

approach	bppv
1st	113
2nd	113
DT3	174
QHULL	288

6 Conclusion

The results show that our second approach yields better timings and memory usage than all the other tested software packages for large data-sets, while being fully robust. This justifies a typical phenomenon: the well-designed specialized solution outperforms the more general one. Here the specialized one is our second approach, and the general one is the Delaunay triangulation 3D computation from which the 3D convex hull is extracted.

The first approach is slower but still one of the most scalable. It exactly computes the triangulation for input points with algebraic coordinates lying on the sphere, and thus ensures that in any case all points will appear in the triangulation. It is the only one to do so within reasonable time and thus being useful for real-world applications.

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.

[11] Memory usage measurements are done with `CGAL::Memory_sizer` for the first approach, second approach and for the 3D Delaunay triangulation. For the Qhull package the measurement is done with the -Ts option, taking into account the memory allocated for facets and their normals, neighbor and vertex sets.

References

1. Hull, a program for convex hulls, http://www.netlib.org/voronoi/hull.html
2. Qhull, http://www.qhull.org/
3. CGAL, Computational Geometry Algorithms Library, http://www.cgal.org
4. CORE number library, http://cs.nyu.edu/exact/core_pages
5. LEDA, Library for efficient data types and algorithms, http://www.algorithmic-solutions.com/enleda.htm
6. Three-dimensional convex hulls, http://www.simplex.t.u-tokyo.ac.jp/~sugihara/opensoft/opensofte.html
7. IEEE standard for floating-point arithmetic. IEEE Std 754-2008, pp. 1–58 (August 2008)
8. Aurenhammer, F.: Power diagrams: properties, algorithms and applications. SIAM Journal of Computing 16, 78–96 (1987)
9. Aurenhammer, F.: Voronoi diagrams: A survey of a fundamental geometric data structure. ACM Computing Surveys 23(3), 345–405 (1991)
10. Berger, M.: The space of spheres. In: Geometry, vol. 1-2, pp. 349–361. Springer, Heidelberg (1987)
11. Boissonnat, J.D., Yvinec, M.: Algorithmic Geometry. Cambridge University Press, UK (1998); Translated by Hervé Brönnimann
12. Bowyer, A.: Computing Dirichlet tessellations. The Computer Journal 24(2), 162–166 (1981)
13. Brown, K.Q.: Geometric transforms for fast geometric algorithms. Ph.D. thesis, Dept. Comput. Sci., Carnegie-Mellon Univ., Pittsburgh, PA, Report CMU-CS-80-101 (1980)
14. Buchin, K.: Constructing Delaunay triangulations along space-filling curves. In: Fiat, A., Sanders, P. (eds.) ESA 2009. LNCS, vol. 5757, pp. 119–130. Springer, Heidelberg (2009)
15. Caroli, M., Teillaud, M.: 3D periodic triangulations. In: CGAL Editorial Board (ed.) CGAL User and Reference Manual, 3.5 edn. (2009)
16. Caroli, M., Teillaud, M.: Computing 3D periodic triangulations. In: Fiat, A., Sanders, P. (eds.) ESA 2009. LNCS, vol. 5757, pp. 37–48. Springer, Heidelberg (2009); Full version available as INRIA Reserch Report No 6823, http://hal.inria.fr/inria-00356871
17. Clarkson, K.L.: Safe and effective determinant evaluation. In: Proceedings 33rd Annual IEEE Symposium on Foundations of Computer Science, October 1992, pp. 387–395 (1992)
18. Clarkson, K.L., Mehlhorn, K., Seidel, R.: Four results on randomized incremental constructions. Computational Geometry: Theory and Applications 3(4), 185–212 (1993)
19. Coxeter, H.S.M.: A geometrical background for de Sitter's world. American Mathematical Monthly 50, 217–228 (1943)
20. de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O.: Computational Geometry: Algorithms and Applications, 2nd edn. Springer, Berlin (2000)
21. de Castro, P.M.M., Cazals, F., Loriot, S., Teillaud, M.: 3D spherical geometry kernel. In: CGAL User and Reference Manual. CGAL Editorial Board, 3.5 edn. (2009)
22. de Castro, P.M.M., Cazals, F., Loriot, S., Teillaud, M.: 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)

23. Delage, C.: Spatial sorting. In: CGAL Editorial Board (ed.) CGAL User and Reference Manual, 3.5 edn. (2009)
24. Devillers, O., Meiser, S., Teillaud, M.: The space of spheres, a geometric tool to unify duality results on Voronoi diagrams. In: Proceedings 4th Canadian Conference on Computational Geometry, pp. 263–268 (1992); Full version available as INRIA Research Report No 1620, http://hal.inria.fr/inria-00074941
25. Fogel, E., Setter, O.: Software for Voronoi diagram on a sphere. Personal communication
26. Fogel, E., Setter, O., Halperin, D.: Exact implementation of arrangements of geodesic arcs on the sphere with applications. In: Abstracts of 24th European Workshop on Computational Geometry, pp. 83–86 (2008)
27. Goldberg, D.: What every computer scientist should know about floating-point arithmetic. ACM Computing Surveys 23(1), 5–48 (1991)
28. Hert, S., Schirra, S.: 3D convex hulls. In: CGAL Editorial Board (ed.) CGAL User and Reference Manual, 3.5 edn. (2009)
29. Kettner, L., Mehlhorn, K., Pion, S., Schirra, S., Yap, C.: Classroom examples of robustness problems in geometric computations. Computational Geometry: Theory and Applications 40, 61–78 (2008)
30. Lawson, C.L.: Software for C^1 surface interpolation. In: Rice, J.R. (ed.) Math. Software III, pp. 161–194. Academic Press, New York (1977)
31. Li, C., Pion, S., Yap, C.K.: Recent progress in exact geometric computation. Journal of Logic and Algebraic Programming 64(1), 85–111 (2005)
32. Muller, J.M.: On the definition of ulp(x). Research Report 5504, INRIA (February 2005), http://hal.inria.fr/inria-00070503/
33. Na, H.S., Lee, C.N., Cheong, O.: Voronoi diagrams on the sphere. Computational Geometry: Theory and Applications 23, 183–194 (2002)
34. Pion, S., Teillaud, M.: 3D triangulation data structure. In: CGAL Editorial Board (ed.) CGAL User and Reference Manual, 3.5 edn. (2009)
35. Pion, S., Teillaud, M.: 3D triangulations. In: CGAL Editorial Board (ed.) CGAL User and Reference Manual, 3.5 edn. (2009)
36. Pion, S., Yvinec, M.: 2D triangulation data structure. In: CGAL Editorial Board (ed.) CGAL User and Reference Manual, 3.5 edn. (2009)
37. Renka, R.J.: 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
38. Sugihara, K.: Laguerre Voronoi diagram on the sphere. Journal for Geometry and Graphics 6(1), 69–81 (2002)
39. Yap, C.K., Dubé, T.: The exact computation paradigm. In: Du, D.-Z., Hwang, F.K. (eds.) Computing in Euclidean Geometry, 2nd edn. Lecture Notes Series on Computing, vol. 4, pp. 452–492. World Scientific, Singapore (1995)
40. Yvinec, M.: 2D triangulations. In: CGAL Editorial Board (ed.) CGAL User and Reference Manual, 3.5 edn. (2009)