Sweeping the Sphere

João Dinis
Departamento de Física
Faculdade de Ciências, Universidade de Lisboa
Campo Grande, Edifício C8
1749-016 Lisboa, Portugal
jd@fc.ul.pt

Margarida Mamede
CITI, Departamento de Informática
Faculdade de Ciências e Tecnologia, FCT
Universidade Nova de Lisboa
2829-516 Caparica, Portugal
mm@di.fct.unl.pt

***Abstract*—We introduce the first sweep line algorithm for computing spherical Voronoi diagrams, which proves that Fortune's method can be extended to points on a sphere surface. This algorithm is similar to Fortune's plane sweep algorithm, sweeping the sphere with a circular line instead of a straight one.**

Like its planar counterpart, the novel linear-space algorithm has worst-case optimal running time. Furthermore, it copes very well with degeneracies and is easy to implement. Experimental results show that the performance of our algorithm is very similar to that of Fortune's algorithm, both with synthetic data sets and with real data.

The usual solutions make use of the connection between convex hulls and spherical Delaunay triangulations. An experimental comparison revealed that our algorithm outperforms the freely available implementations that compute convex hulls of point sets in 3D, enabling it to be the preferred choice for computing Voronoi diagrams on the sphere.

***Keywords*-spherical Voronoi diagrams; sweep algorithms.**

I. Introduction

Voronoi diagrams are one of the most important and useful geometrical data structures that find use in a variety of fields. Originally defined for point sites in the plane, where proximity is defined by the Euclidean distance, they were soon generalized to different site shapes, spaces, and metrics [1], [2].

One possible generalization, called *spherical Voronoi diagram*, is cast by considering point sites on the surface of a sphere and measuring the proximity of two points through the length of the shortest geodesic arc that joins them [3]. It is similar to the planar diagram, except that all Voronoi regions are delimited by spherical polygons, whose edges are arcs of great circles. These Voronoi diagrams are of special interest in domains where data are inherently spherical, such as Earth sciences and astronomy.

The first algorithm for computing the spherical Voronoi diagram makes use of the connection between convex hulls and spherical Delaunay triangulations [4], because a facet of the convex hull corresponds to a triangle of the Delaunay triangulation and, therefore, to a vertex of the Voronoi diagram. Several algorithms may be used for that purpose. Brown [4] suggests the divide and conquer algorithm of Preparata and Hong [5], [6], which is $O(n \log n)$ worst case optimal, where n is the number of sites. An asymptotically slower alternative in the worst case is the randomized incremental algorithm of Clarkson and Shor [7], whose expected running time is $O(n \log n)$. The well-known Quickhull algorithm of Barber *et al.* [8] can be seen as an efficient variation of the previous algorithm.

A different approach is to adapt the randomized incremental algorithm that computes the planar Delaunay triangulation [9], [10]. The essential operation used in the incremental step, which checks if a circle defined by three sites contains a fourth site, is easily translated into spherical geometry [11], [12]. This algorithm computes the Delaunay triangulation in $O(n \log n)$ expected time.

The two incremental methods are comparable, since the corresponding incremental steps require a similar effort to recompute the part invalidated by the newly added site. This is a consequence of the combinatorial equivalence between convex hulls and Delaunay triangulations of points on a sphere surface.

Another important result, also due to Brown [4], relates the points on the sphere with their stereographic projections on a plane. It is worth mentioning that, whereas Brown made use of this property to build Voronoi diagrams in the plane, Na *et al.* [13] relied on the inverse relationship and showed how to compute the spherical Voronoi diagram by combining two planar Voronoi diagrams of the projected sites. The latter result works for any compact sites. For point sites, the running time of the algorithm is $O(n \log n)$.

In this work, we study the construction of Voronoi diagrams of point sites on the surface of a sphere. We introduce a novel algorithm based on the sweep line technique, using an approach similar to that of Fortune for the planar case [14]. More concretely, the spherical Voronoi diagram will be computed by sweeping the sphere with a circular line. This strategy was previously used by Dehne and Klein [15] to build the Voronoi diagram on a cone surface. However, we shall adopt the variant of the sweep line algorithm of Guibas and Stolfi [16], preferring the now common wave front construct so as to avoid the deformation transform.

In the plane, the sweep line algorithm is consensually

regarded as computationally simple, coping very well with degeneracies, and easy to implement. As we will see, all these properties also hold in the spherical version.

The rest of the paper is organized as follows. The spherical sweep algorithm is specified in Section II. Section II-A starts examining spherical ellipses, which are needed to define the wave front, studied in Section II-B. Then, in Section II-C the algorithm is presented and analyzed, and degenerated cases are discussed in Section II-D. Although the algorithm is defined for the spherical coordinate system, Section II-E is focused on the details of working with Cartesian coordinates. Section III reports on the experimental results: our algorithm is compared, in Section III-A, with two sweep line algorithms for computing Voronoi diagrams in the plane and, in Section III-B, with five freely available implementations for computing convex hulls in 3D. Lastly, Section IV concludes with some comments on the research done in the paper.

II. Spherical Sweep

This section is devoted to the algorithm for computing the Voronoi diagram on a sphere. It turns out to be an almost straightforward adaptation of the circular sweep algorithm in the plane [15].

Due to its convenience, we make use of the standard terminology in geography, namely, latitude, colatitude, longitude, north and south poles, parallel of latitude, meridian of longitude, prime meridian, and antipode of a site.

First of all, an arbitrary sphere position is chosen to be the *center of the sweep*. The sweep circle is always centered at that position and its radius, which is the length of a geodesic arc, starts at zero and increases as the sweep process advances. When a site is scanned by the sweep circle, we are interested in the loci of points equidistant to the site and the circle, which form a *spherical ellipse*. The outer envelope of all these ellipses constitutes the *wave front*. It is a closed curve made of arcs of ellipse whose intersections scan the edges of the Voronoi diagram. In this case, since the sphere is a closed domain, the wave front disappears when the sweep process comes to an end.

Although the center of the sweep may be placed at any location, we choose the north pole because it greatly simplifies the analysis.[1] As a consequence, the sweep circle scans the parallels of latitude, and its radius is equal to the colatitude of the parallel. The key fact is that the sweep circle needs to pass over the sphere twice: first from the north pole to the south pole, and then backwards to the north.

A. Spherical Ellipses

Since the wave front is made of arcs of ellipse, we will start by analyzing the properties of an ellipse generated by the sweep circle and a site, and how it sweeps the sphere.

Let $a = (a_\rho, a_\theta)$ be a site on the sphere, where a_ρ is the colatitude and a_θ is the longitude. Let also $o = (0, 0)$ be the center of the sweep, and $r \geq a_\rho$ be the radius of the sweep circle. The loci of points i equidistant to a and to the circle define a spherical ellipse, with o and a as focal points and r as major axis (see Fig. 1). This follows from the fact that, for every i in the conditions stated above, there is a point p on the circle such that $\overline{p\ i} = \overline{i\ a}$ and $\overline{o\ i} + \overline{i\ p} = r$. Therefore, $\overline{o\ i} + \overline{i\ a} = r$. The equation of the spherical ellipse, which gives the colatitude (i_ρ) of the point of the curve whose longitude is i_θ, is the following (where the $\arctan$ function is normalized for positive output):

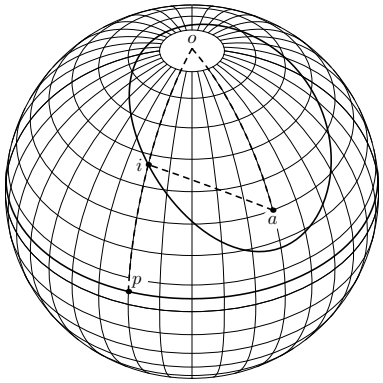

Figure 1. Spherical ellipse defined by a circle and site a.

$$\mathcal{E}(a, r, i_\theta) = \arctan\left(\frac{\cos(a_\rho) - \cos(r)}{\sin(r) - \sin(a_\rho) \cdot \cos(a_\theta - i_\theta)}\right). \quad (1)$$

Equation (1) shows that the ellipse major axis is aligned with the site meridian. The ellipse begins degenerated as an arc of great circle delimited by the foci points, and widens as r increases. When r reaches π, the ellipse degenerates into a great circle, because the sweep circle is the south pole and the ellipse is the bisector of two points on the sphere. At that moment, the ellipse has swept half-sphere.

The next proposition proves that, since half of the sphere is swept when r varies from a_ρ to π, the remaining half is swept for r from π to $2\pi - a_\rho$. Furthermore, the two sweep phases are symmetric, in the sense that the ellipse generated for a sweep circle at $\pi + \delta$ is the mirror image of the ellipse generated for the sweep circle at $\pi - \delta$, taking the great circle obtained when $r = \pi$ as the mirror.

Proposition 1: Let $a = (a_\rho, a_\theta)$ be a site, o be the north pole, a^* and o^* be the antipodes of a and o, respectively, and $\delta \in [0, \pi - a_\rho]$. The spherical ellipse generated by site a when the sweep circle has radius $\pi + \delta$ is the ellipse with foci a^* and o^* whose major axis is $\pi - \delta$.

Proof: Let i be a point of the ellipse generated by a when the sweep circle has radius $\pi + \delta$. Then, $\overline{i\ a} + \overline{i\ o} = \pi + \delta$. Therefore, $\overline{i\ a^*} + \overline{i\ o^*} = (\pi - \overline{i\ a}) + (\pi - \overline{i\ o}) = 2\pi - (\pi + \delta) = \pi - \delta$. Hence, i is on the ellipse with foci a^* and o^* whose major axis is $\pi - \delta$. ■

To sum up, the ellipse starts degenerated as an arc

[1] Any other location would be equivalent after a proper rotation.

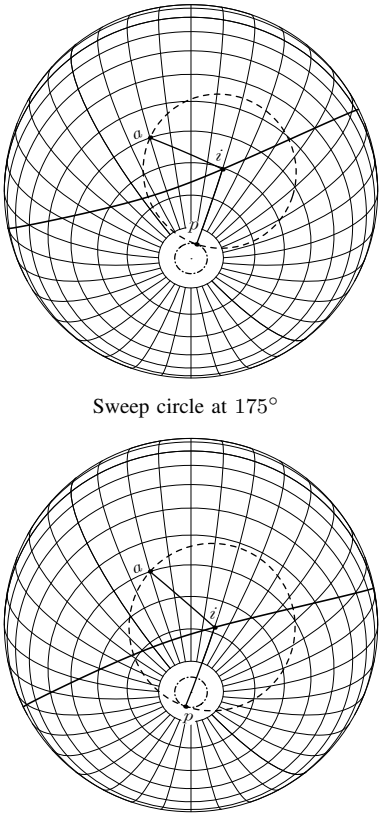

Figure 2. Crossing the south pole. The solid line represents the visible part of the ellipse. The dashed line is the largest empty circle for a point on the ellipse. The dashed-dot line is the sweep circle.

connecting the foci, widens until it degenerates into a great circle, and then narrows until it ends as an arc linking the foci antipodes. Moreover, the ellipse effectively sweeps all the sphere, when the circle radius varies from a_ρ to $2\pi - a_\rho$.

The sweep process has an interesting geometric interpretation, regarding maximum empty circles (see Fig. 2). The largest empty circle centered at a point i of the ellipse is tangent to site a and to a point p of the sweep circle with radius r. If $r < \pi$, the largest empty circle is tangent to the upper part of the sweep circle (on the side of the smallest colatitudes) and does not contain the south pole. For $r > \pi$, the largest empty circle is tangent to the lower part of the sweep circle (on the side of the greatest colatitudes) and contains the south pole. When $r = \pi$, the sweep circle coincides with the south pole, to which the largest empty circle is tangent.

B. Wave Front

The wave front is the lower envelope of the ellipses generated by the sites already scanned by the sweep circle. It begins its existence when the sweep circle scans the northernmost site, starting degenerated as an arc of great circle. The sweep process ends when the sweep circle

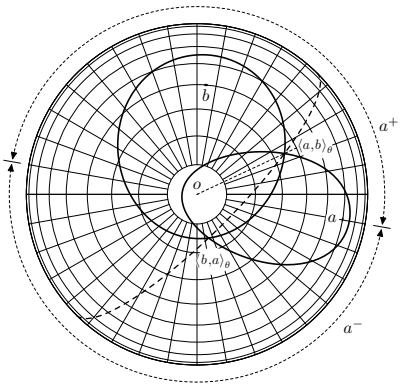

Figure 3. Intersection of two spherical ellipses.

rescans the first site after the south pole, that is, when it scans the southernmost site a second time. At that moment, the wave front coincides with the ellipse generated by that site, which is again an arc of great circle.

Our next goal is to understand how spherical ellipses intersect. To start with, we define an order relation on sites by:

$$a < a' \;\Leftrightarrow\; a_\rho < a'_\rho \vee (a_\rho = a'_\rho \wedge a_\theta < a'_\theta).$$

Let $a = (a_\rho, a_\theta)$ and $b = (b_\rho, b_\theta)$ be two distinct sites such that $b < a$, $r \in [a_\rho, 2\pi - a_\rho]$ be the radius of the sweep circle, and o be the north pole (as illustrated in Fig. 3). The ellipses generated by a and b intersect at exactly two points, named $\langle a,b\rangle$ and $\langle b,a\rangle$, which, in this case, are on opposite sides of the great circle defined by the a_θ meridian (because b is not farther from o than a). Therefore, $\langle a,b\rangle_\theta \in a^+$ and $\langle b,a\rangle_\theta \in a^-$, where a^+ and a^- are the hemispheres defined by the meridians of longitude $]a_\theta, a_\theta + \pi[$ and $]a_\theta - \pi, a_\theta[$, respectively.

The two arc intersections start out coincident, when $r = a_\rho$. Then, they scan the sites bisector following in opposite directions: $\langle a,b\rangle_\theta$ monotonically increases in θ whereas $\langle b,a\rangle_\theta$ monotonically decreases in θ. When r reaches $2\pi - a_\rho$, the two intersections coincide again, completing the scan of the bisector. Notice that, if $b_\rho = a_\rho$, the sites bisector is a meridian, so $\langle a,b\rangle_\theta$ and $\langle b,a\rangle_\theta$ are constant along the sweep. However, to avoid special cases, it may be assumed that $\langle a,b\rangle_\theta$ (respectively, $\langle b,a\rangle_\theta$) effectively sweeps a^+ (respectively, a^-).

Similar results are obtained for $a < b$.

C. The Algorithm

The spherical sweep algorithm does not differ much from the circular sweep algorithm in the plane [15]. A way to cope with the closed nature of the wave front, which is ordered in θ, is to cut it by the prime meridian and to represent the divided arc twice, in the beginning and in the end of the sequence of arcs and arc intersections. Whenever one of the arc intersections traced by the duplicated arc crosses the

prime meridian, a roll-event takes place. Events are stored in an adaptable priority queue [17], ordered by colatitude (which may vary from 0 to 2π).

The algorithm starts by inserting all sites into the priority queue. Then, while the priority queue is not empty, an event is removed and processed. When the queue is exhausted, a last step must be performed. Since the event processing is the usual one, in what concerns additions and removals of arcs and arc intersections, we will restrict ourselves to the scheduling of roll- and circle-events.

A roll-event is associated with the first arc $\langle a \rangle$ of the wave front if the first arc intersection $\langle a,b \rangle$ crosses the prime meridian. This happens only if $b > a$ and $b_\theta < \pi$. Similarly, a roll-event is associated with the last arc $\langle d \rangle$ of the wave front when the penultimate arc $\langle c \rangle$ is such that $c > d$ and $c_\theta > \pi$. The priority of a roll-event is determined by computing the circle centered on the prime meridian that circumscribes the two corresponding sites.

Basically, a circle-event should be scheduled if the paths of the adjacent arc intersections intersect before any of them crosses the prime meridian. Let $\langle b \rangle$ be an arc, and $\langle a,b \rangle$ and $\langle b,c \rangle$ be the two adjacent intersections. Let also $\langle a,b,c \rangle$ be the center of the spherical circle that circumscribes, in clockwise order, sites a, b, and c. If $a < b$ and $b > c$, the two arc intersections $\langle a,b \rangle$ and $\langle b,c \rangle$ run apart, so arc $\langle b \rangle$ does not disappear. Any other case may lead to a circle-event associated with $\langle b \rangle$, which will be scheduled if one of the following conditions is met.

- $a > b$ and $b < c$
  In this case, the two intersections run against each other and $\langle a,b,c \rangle_\theta \in [a_\theta, c_\theta]$.
- $a > b$, $b > c$, and $\langle a,b,c \rangle_\theta \in a^+ \cap b^+ \cap]a_\theta, 2\pi[$
  Both arc intersections (which scan points of increasing longitude) must meet within their ranges and before any of them crosses the prime meridian.
- $a < b$, $b < c$, and $\langle a,b,c \rangle_\theta \in b^- \cap c^- \cap [0, c_\theta[$
  This is the opposite of the previous case.

The priority of the circle-event is the greater colatitude of a point of the circumference defined by the three sites.

There is an additional global constraint for a circle- or a roll-event to be scheduled: its priority cannot exceed $2\pi - s_\rho$, where s is the southernmost site, as the sweep process ends when the sweep radius reaches that value.

It is important to mention that the processing of circle-events always generates vertices of degree three. Actually, vertices of greater degree appear in the resulting diagram as more than one vertex, overlapped and connected by null length edges. So, an immediate consequence of Euler's formula is that a Voronoi diagram of n sites built by our algorithm has $2\,n - 4$ vertices and $3\,n - 6$ edges (for any $n \geq 3$).

Proposition 2: For $n \geq 2$ sites, the wave front ends with two arcs (and two intersections).

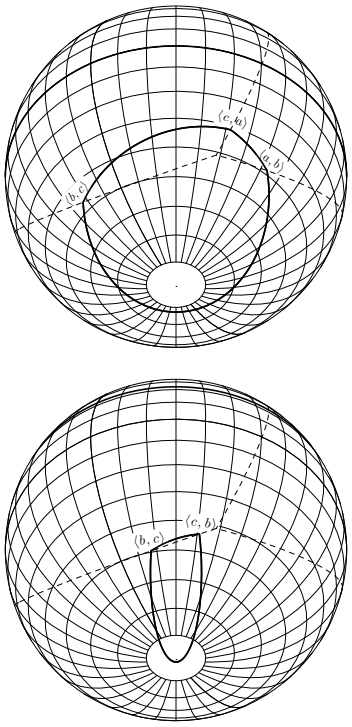

Figure 4. Closing the wave front, before and after the last circle-event.

Proof: The first two site-events give rise to exactly two arcs and two intersections in the wave front (seen as a closed line). If $n = 2$, the wave front does not change anymore. For $n > 2$, each of the subsequent site-event adds two arcs to the wave front, while each circle-event subtracts one arc. Therefore, the arcs added by $n - 2$ site-events are offset by the arcs subtracted by $2\,n - 4$ circle-events. ■

The two remaining intersections of the wave front must scan the same edge of the Voronoi diagram (see Fig. 4). Otherwise, at least one vertex (and a circle-event) would be missing. This edge, which is being scanned twice, has also been added twice to the diagram. Therefore, the diagram is completed by discarding one of the edges and appropriately connecting the other one.

Now, let us define the ordering relation between an arbitrary longitude θ' and an arc intersection $\langle a,b \rangle$, used in the binary search tree that implements the wave front. When $a > b$, $\langle a,b \rangle_\theta \in a^+$, and three distinct cases (illustrated in Fig. 5) must be analyzed.

- If the prime meridian $\notin a^+$, the sphere may be divided into three sectors: $[0, a_\theta]$, a^+, and $[a_\theta + \pi, 2\pi[$. Notice that only a^+ poses some difficulties, because, if θ' belongs to the first or to the third sectors, it is lower or greater than $\langle a,b \rangle_\theta$, respectively. The comparison in a^+ makes use of (1):

$$\theta' \leq \langle a,b \rangle_\theta \;\; \Leftrightarrow \;\; \mathcal{E}(a, r, \theta') \geq \mathcal{E}(b, r, \theta'). \qquad (2)$$

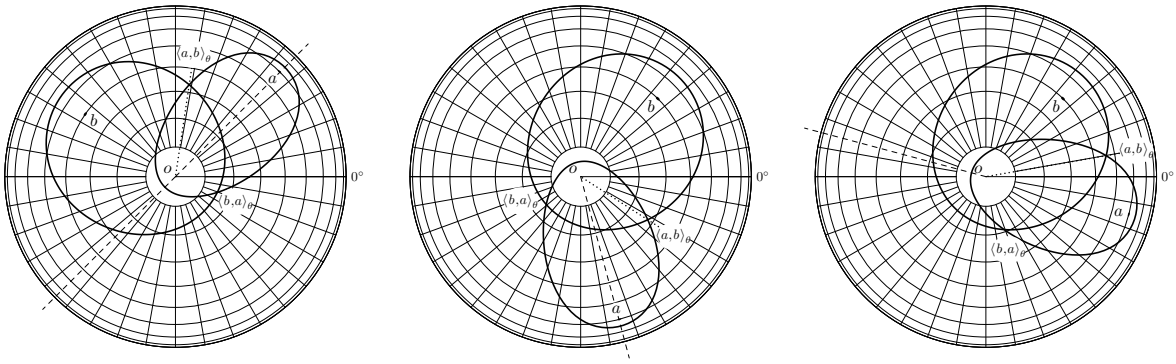

Figure 5. Comparing a longitude with $\langle a,b\rangle$, when $a > b$.

- When a^+ contains the prime meridian, two situations are considered.
  - If $\langle a,b\rangle_\theta \in [a_\theta, 2\pi[$, either $\theta' \in [0, a_\theta]$, which implies $\theta' \leq \langle a,b\rangle_\theta$, or $\theta' \in]a_\theta, 2\pi[$. The latter case is solved by (2).
  - When $\langle a,b\rangle_\theta \in [0, a_\theta - \pi]$, if $\theta' \in [a_\theta - \pi, 2\pi[$, then $\theta' \geq \langle a,b\rangle_\theta$; otherwise, $\theta' \in [0, a_\theta - \pi[$ and (2) is used.

The cases in which $a < b$ can be treated in a similar way.

Proposition 3: The algorithm computes the spherical Voronoi diagram with n sites in $O(n \log n)$ time using $\Theta(n)$ space.

Proof: First of all, the size of both data structures is $\Theta(n)$. Therefore, each primitive operation on the binary search tree and on the priority queue takes $O(\log n)$ steps, if they are implemented, respectively, with a red-black tree [18] and with a binary heap and an array.[2] Besides, every event requires a constant number of those primitive operations to be processed, and the total number of processed events is $\Theta(n)$. ■

A consequence of Brown's result [4] that links Voronoi diagrams of sites on a sphere with Voronoi diagrams of their stereographic projections on a plane is that any algorithm for computing the spherical Voronoi diagram implicitly computes two planar Voronoi diagrams: a nearest point diagram and a furthest point diagram. It is easy to see that the spherical sweep algorithm computes them in turn. Consider the stereographic projection center c at the south pole. The nearest diagram is computed when the sweep circle radius varies from 0 to π, and the maximum empty circle of every vertex does not include c. Then, the furthest diagram is computed when the sweep circle radius varies from π to 2π, and the maximum empty circle of any vertex includes c.

[2]The second array associates every event in the priority queue with its position in the binary heap.

D. Site at the South Pole

The algorithm described above is well behaved for sites placed anywhere on the sphere, except at the south pole. Let s be a site at the south pole. To start with, the valid range of the sweep circle radius is the degenerated interval $\{\pi\}$. Moreover, the initial ellipse, which is always an arc connecting the foci points, is indeterminate, because any meridian fulfills the requirements.

In order to see that this problem does not compromise the algorithm, recall that, when the sweep circle reaches the south pole, every arc $\langle a\rangle$ of the wave front is an arc of great circle, which bisects the corresponding site a and s. As a matter of fact, the Voronoi polygon of site s coincides with the wave front when the sweep circle reaches π. The justification comes from Proposition 4.

Proposition 4: When the sweep circle radius is π, any two distinct arcs of the wave front (seen as a closed line) are generated by distinct sites.

Proof: The proof is straightforward when the wave front has only one or two arcs. So, let us assume that $\langle a\rangle$, $\langle b\rangle$, and $\langle c\rangle$ are three consecutive arcs of the wave front, generated by sites a, b and c, respectively, separated by intersections $\langle a,b\rangle$ and $\langle b,c\rangle$. Our goal is to show that b cannot generate any arc but $\langle b\rangle$. Since $\langle a,b\rangle$ is the intersection of two great circles (and the wave front is the lower envelope of arcs), site b cannot generate any arc in the interval $]\langle a,b\rangle_\theta - \pi, \langle a,b\rangle_\theta[$, where the great circle generated by $\langle a\rangle$ has a greater colatitude than the great circle generated by $\langle b\rangle$. Similarly, b cannot generate any arc in the interval $]\langle b,c\rangle_\theta, \langle b,c\rangle_\theta + \pi[$, where the great circle generated by $\langle c\rangle$ has a greater colatitude than the great circle generated by $\langle b\rangle$. As the union of both intervals corresponds to $[0, 2\pi]\backslash]\langle a,b\rangle_\theta, \langle b,c\rangle_\theta[$, $\langle b\rangle$ is the only arc generated by b. ■

In practice, events due to a site at the south pole do not need a special treatment. When the site-event is handled, an arc of the wave front is arbitrary chosen to be the one directly above the south pole and the corresponding new arc

starts. This new arc gives rise to a cascade of circle-events, all scheduled for colatitude π, whose processing computes the vertices of the south pole Voronoi polygon. It is irrelevant which arc is chosen to be directly above the south pole. So, the single modification from the general case is to replace the computation of $\frac{0}{0}$ with any value in (1).

We conclude that sites can be positioned anywhere on the sphere, even though the sweep process ends rather abruptly if a site is placed at the south pole.

E. Spherical and Cartesian Coordinate Systems

The actual computation of event priorities and the definition of the ordering relation used in the wave front depend on the underlying system of coordinates. We will now focus on the details for the two common alternatives.

So far, we have adopted the spherical coordinate system, since it is the natural choice when dealing with locations on a sphere. In this setting, a site a is defined by its colatitude a_ρ and its longitude a_θ, thus the priority of the corresponding site-event is

$$P^s_{\text{site}}(a) = a_\rho. \tag{3}$$

The priority of a circle-event requires the computation of the circle circumscribed by three sites, a, b and c, which is defined by the intersection of the plane containing the sites and the unit sphere. If $\vec{p} = (p_x, p_y, p_z)$ denotes (the vector with) the Cartesian coordinates of a point p on the sphere, the priority of the circle-event is given by:

$$P^s_{\text{circle}}(a, b, c) = \arccos(u_z) + \arccos(\vec{u} \cdot \vec{a}), \tag{4}$$

where

$$\begin{cases} \vec{v} = (\vec{a} - \vec{b}) \times (\vec{c} - \vec{b}), \\ \vec{u} = \frac{\vec{v}}{\|\vec{v}\|}. \end{cases}$$

The circle-event should be scheduled if, and only if, the corresponding intersections $\langle a,b\rangle$ and $\langle b,c\rangle$ cross at the circle center u. Because this verification is made by comparing u_θ with some values in the set $\{0, 2\pi, a_\theta, a_\theta \pm \pi, b_\theta, b_\theta \pm \pi, c_\theta, c_\theta \pm \pi\}$, rounding errors may prevent true circle-events to be scheduled. This difficulty can be overcome by scheduling all circle-events, except those whose arc intersections run apart. Of course this strategy may schedule some extra circle-events, which will be later identified as false events, but none of the true events is left out.

The priority of a roll-event caused by an arc intersection $\langle a,b\rangle$ is computed in a similar way:

$$P^s_{\text{roll}}(a, b) = \arccos(u_z) + \arccos(\vec{u} \cdot \vec{a}), \tag{5}$$

where

$$\begin{cases} \vec{v} = (a_z - b_z, 0, b_x - a_x), \\ \vec{u} = \frac{\vec{v}}{\|\vec{v}\|}. \end{cases}$$

In this case, the decision of scheduling the event depends solely on the comparison of a_θ (or b_θ) with π.

The ordering relation between a longitude θ' and an arc intersection $\langle a,b\rangle$ has already been completely detailed in Section II-C.

A straightforward implementation of the previous definitions has two major drawbacks: it relies on comparisons with inexact values (e.g. π) and requires the use of trigonometric functions. That is why Cartesian coordinates are preferable.

Let sites be specified in (or converted to) Cartesian coordinates. A priority associated with the sweep circle at radius ρ is:

$$P^c(\rho) = 2\,\text{sgn}(t)\,t^2, \text{ where } t = \cos(\rho/2). \tag{6}$$

Now, priorities vary from $+2$ to -2, are positive when the sweep circle moves southwards, zero at the south pole, and negative while the sweep circle goes toward the north. Applying (6) to the three types of events gives, after simplifying expressions, the following results.

The priority of a site-event is simply:

$$P^c_{\text{site}}(a) = 1 + a_z. \tag{7}$$

The priority of a circle-event is given by:

$$P^c_{\text{circle}}(a, b, c) = \frac{\delta + r\,v_z - \sqrt{(\delta - v_z^2)(\delta - r^2)}}{\delta}, \tag{8}$$

where

$$\begin{cases} \vec{v} = (\vec{a} - \vec{b}) \times (\vec{c} - \vec{b}), \\ r = \vec{v} \cdot \vec{a}, \\ \delta = \|\vec{v}\|^2. \end{cases}$$

The priority of a roll-event is:

$$P^c_{\text{roll}}(a, b) = s \times \frac{\delta + r\,z - \sqrt{(\delta - z^2)(\delta - r^2)}}{\delta}, \tag{9}$$

where

$$\begin{cases} r = \text{sgn}(b_z - a_z)\ (a_x b_z - a_z b_x), \\ z = \text{sgn}(b_z - a_z)\ (a_x - b_x), \\ s = \text{sgn}(r + z), \\ \delta = (a_x - b_x)^2 + (a_z - b_z)^2. \end{cases}$$

The ordering relation used in the wavefront also becomes simpler. The comparison of a direction, defined by a site p, with an intersection $\langle a,b\rangle$ verifies:

$$p_\theta < \langle a,b\rangle_\theta \Leftrightarrow \alpha\,|b_z - p_z| < \beta\,|a_z - p_z|, \tag{10}$$

where

$$\begin{cases} \alpha = p_x\ (p_x - a_x) + p_y\ (p_y - a_y), \\ \beta = p_x\ (p_x - b_x) + p_y\ (p_y - b_y). \end{cases}$$

Therefore, the use of Cartesian coordinates together with a redefinition of priority enables to implement the spherical sweep algorithm employing only arithmetic and square root operations.

III. Experimental Results

In this section, we present some experimental results that show how the spherical sweep algorithm behaves and how it compares with other algorithms. In Section III-A, the study is restricted to sweep algorithms and, in Section III-B, it is centered on algorithms for computing spherical Voronoi diagrams.

A. Planar and Spherical Sweep Algorithms

We have programmed four versions of sweep algorithms. The *linear* [14] and the *circular* [15] sweep algorithms compute Voronoi diagrams in the plane. The first one scans the plane with a straight line, whereas the second uses a circular line. In both cases, points are defined in Cartesian coordinates. Besides, there are two implementations of the spherical sweep algorithm: one called *spherical s*, for sites in spherical coordinates, which makes use of trigonometric functions; and the other named *spherical c*, where points are in Cartesian coordinates.

Whenever possible, source code was shared among the implementations so that running times could be comparable. In particular, the relevant data structures are the same across implementations. The binary search tree has been implemented with a red-black tree, and the adaptable priority queue with a binary heap and an array. The main implementation differences rely on computing priorities and keeping the wave front ordered.

All algorithms have been coded in the C language and compiled with the GNU GCC C compiler (version 4.3.2) with optimizations. Programs have been executed on an Intel Xeon E5160 processor, running at 3.00 GHz, with memory running at 1333 MHz.

With respect to data sets, sites on the plane are at random positions, distributed uniformly on the $[0,1] \times [0,1]$ square. Circular sweeps have been centered at the origin. On the sphere, two types of data sets have been generated. In the first one, sites are also at random positions, uniformly distributed, while, in the second, they belong to the Tycho-2 star catalog [19]. This catalog has over 2.5×10^6 stars, with a visible non-uniform distribution on the celestial sphere. Each Tycho-2 data set has been computed by taking the first stars from a random permutation of all catalog. The star catalog has been chosen in order to test the spherical sweep algorithm with real data.

For each one of those universes, five independent data sets have been generated, with sizes $0.1 \times 10^6, 0.2 \times 10^6, 0.3 \times 10^6, \ldots, 2.5 \times 10^6$. All results presented in the paper are the average of the five values obtained from the executions with the sets of the same nature and size.

To begin with, Fig. 6 plots the average number of false events per site. With respect to this metric, values are equal in both implementations of the spherical sweep. Needless to say, the total number of false events must depend linearly on the number n of sites, because there are $\Theta(n)$ events generated and $\Theta(n)$ events processed. The results show that, when the algorithms are executed with random data sets, the number of false events per site converges to a very similar constant as n increases. Therefore, the costs due to false events are equivalent in all algorithms.

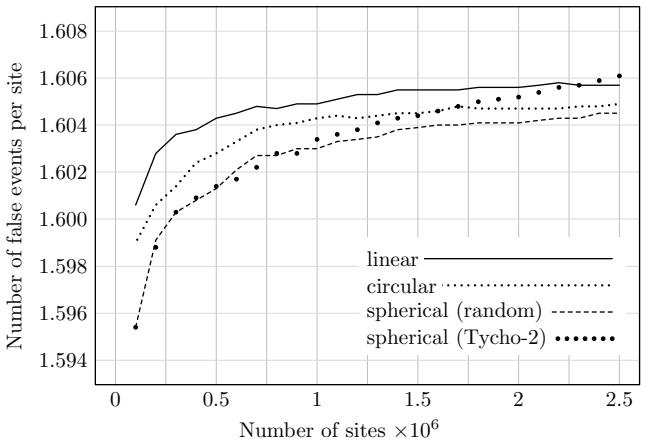

Figure 6. Average number of false events per site.

Fig. 7 depicts running times. All curves exhibit a supra linear behavior, as expected, and, what is more meaningful, they are related with each other by almost "constant" factors. The spherical c, the circular, and the spherical s programs differ from the linear one approximately by a factor of 1.24, 1.33, and 3.18, respectively. The observed patterns are explicable by the costs of computing priorities and keeping the wave front ordered. This is particularly true when comparing the spherical implementations, whose only differences are those stated in Section II-E. Remark that the number of events cannot justify the distinct running times given that, in all algorithms, the numbers of site-events are equal, the numbers of circle-events are approximately equal, the numbers of roll-events (which are very close to $\sqrt{n}$) are

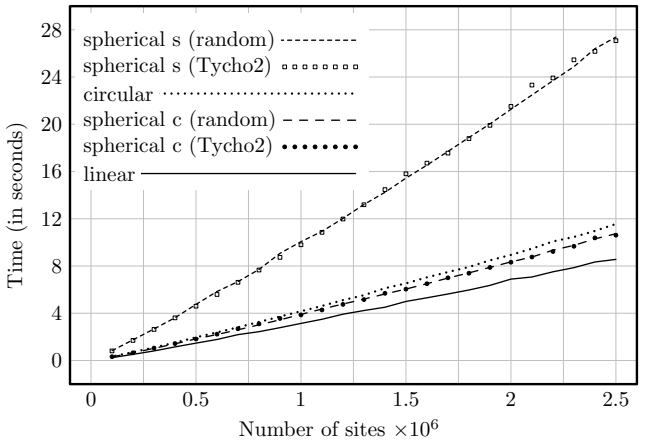

Figure 7. Average running times of sweep algorithms.

insignificant, and the numbers of false events are identical.

It also turns out that the running times with Tycho-2 data sets are almost coincident with those with random data sets. So, at least in this experiment, the non-uniformity has not produced sensible variations.

In conclusion, these experimental results show that the spherical sweep algorithm is not only practical to implement, but also as efficient as Fortune's algorithm.

B. Spherical and 3D Algorithms

Our next goal is to compare the spherical sweep algorithm with five freely available alternatives:

- *Hull* [20] and *CGAL hull* [21], which implement the incremental algorithm of Clarkson [22] for computing the convex hull of a set of points in any dimension;
- *Qhull* [23] and *CGAL quickhull* [21], which implement the Quickhull algorithm of Barber *et al.* [8]; and
- *CGAL triang.* [21], which computes a basic triangulation [24] of a set of points whose dimension is at most three. (Remark that any 3D triangulation of a point set on a sphere also provides its convex hull.)

It is important to mention that [25] presents an adaptation, to the spherical domain, of the incremental algorithm for computing the 2D Delaunay triangulation. It is reported to be faster than that for computing the 3D triangulation. Nevertheless, it is not yet available in the CGAL library. Moreover, the LEDA library [26] has been left out because the free edition does not provide the computation of 3D convex hulls.

All freely available implementations have been compiled and executed as described in Section III-A. Concerning the CGAL library, we followed the conventional choice of using the exact computational kernel and the floating-point data representation. Moreover, we have used the same data sets with points on a sphere.

Since there are significant differences in the running times of the tested implementations, we present them separately. Fig. 8 depicts the running times of the three slower algorithms with random data sets, whereas Fig. 9 plots the corresponding results for the four fastest ones. Hull occurs in both so as to have an easy reference mark. Besides, Fig. 10 presents the running times of the fastest algorithms with Tycho-2 data sets.

CGAL quickhull, CGAL hull, and Hull belong to the slowest group. While the last two are not a surprise, we sought an explanation for the performance of CGAL quickhull, which might be related to the numerical filters of the exact computational kernel that are used to assure that determinant signs are correct. Unfortunately, the adoption of an inexact kernel is not a solution, because the program does not terminate without filtered computations (even with as few as five points).

When we restrict ourselves to the second group, Hull is the slowest algorithm. But is uses precise integer arithmetic,

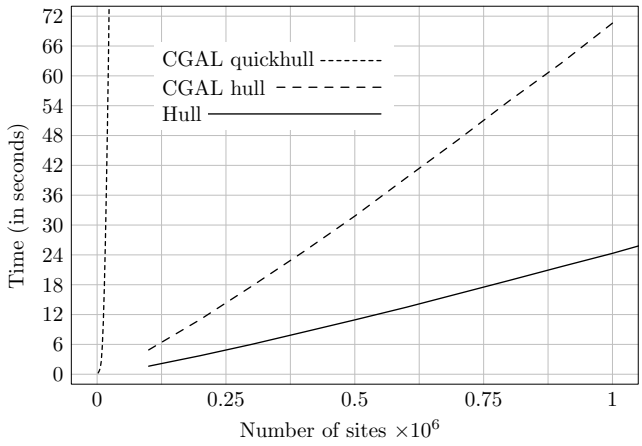

Figure 8. Average running times of the slowest algorithms with random data sets.

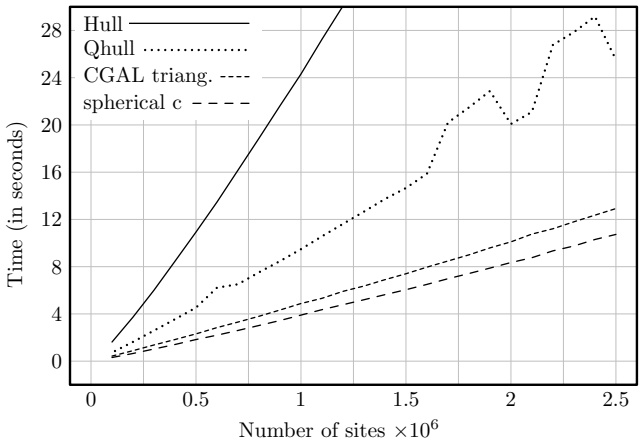

Figure 9. Average running times of the fastest algorithms with random data sets.

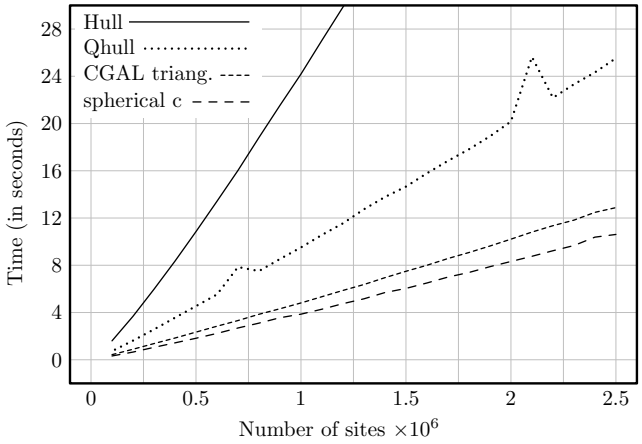

Figure 10. Average running times of the fastest algorithms with Tycho-2 data sets.

evaluates the sign of determinants exactly [27] and, unlike the alternatives, has no improvements to speed up point locations.

Qhull comes in second place. It benefits from the use of floating-point arithmetic and from the efficiency of the Quickhull strategy (where outside sets obviate the need to perform point locations). Nevertheless, it might be affected by all sites being part of the resulting convex hull, since a lot of bookkeeping takes place to maintain the outside sets before a new site is added. Remark that, as our purpose was to obtain a Voronoi diagram on a sphere, we turned on the flag that enforces the output to be a triangulation. That is, degenerated facets, defined by more than three sites, have also been triangulated. Qhull performance is similar to that of our algorithm with spherical coordinates.

The running times of CGAL triang. are only slightly greater than those of the spherical sweep algorithm with Cartesian coordinates. As far as we understood, the good performance of this implementation is mainly due to sites being spatially sorted along a Hilbert curve [28]. Consequently, each new site is close to the last one added, which minimizes, on average, the cost of the walk performed in the current triangulation to find a new cell that contains a site.

Comparing the results obtained with random data and with stars from the Tycho-2 catalog, it turns out that the corresponding curves are almost equal, except those concerning Qhull. The reason is that Qhull running time increases considerably when sites are found very close to the convex hull boundary (and thick facets are built).

In conclusion, these experimental results show that the spherical sweep algorithm is the preferred choice for computing Voronoi diagrams on a sphere.

IV. Conclusions

We have presented a novel algorithm for computing the Voronoi diagram of point sites on a sphere surface, which is an adaption of the sweep technique to the spherical domain. Actually, our algorithm does not mimic closely the linear plane sweeps of Fortune's algorithms [14], but it imitates the round sweep algorithm of Dehne and Klein [15] instead, which proved to be adjustable to the round nature of the sphere.

Our algorithm computes the spherical Voronoi diagram in $O(n \log n)$ time and $\Theta(n)$ space (where n is the number of sites), which is worst-case optimal. Furthermore, the experimental results attested that it is as practical, efficient, and easy to implement as Fortune's algorithm, enabling it to be the preferred choice for computing Voronoi diagrams on a sphere.

An important achievement of this work is the neat adaptation of the circular planar sweep technique to the sphere. This shows that other planar sweep algorithms may be adapted to the spherical domain, as long as their sweep strategy can rely on a circle.

References

[1] F. Aurenhammer, "Voronoi diagrams — a survey of a fundamental geometric data structure," *ACM Computing Surveys*, vol. 23, no. 3, pp. 345–405, 1991.

[2] F. Aurenhammer and R. Klein, "Voronoi diagrams," in *Handbook of Computational Geometry*, J. Sack and G. Urrutia, Eds. Elsevier Science Publishing, 2000, pp. 201–290, [SFB Report F003-092, TU Graz, Austria, 1996].

[3] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, 2nd ed. Chichester: John Wiley, 2000.

[4] K. Q. Brown, "Geometric transforms for fast geometric algorithms," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, USA, 1979.

[5] F. P. Preparata and S. J. Hong, "Convex hulls of finite sets of points in two and three dimensions," *Communications of the ACM*, vol. 20, no. 2, pp. 87–93, 1977.

[6] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*. New York, NY: Springer-Verlag, 1985.

[7] K. L. Clarkson and P. W. Shor, "Applications of random sampling in computational geometry, II." *Discrete and Computational Geometry*, vol. 4, no. 1, pp. 387–421, 1989.

[8] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Transactions on Mathematical Software*, vol. 22, no. 4, pp. 469–483, 1996.

[9] L. J. Guibas and J. Stolfi, "Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams," in *STOC '83: Proceedings of the fifteenth annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, 1983, pp. 221–234.

[10] L. J. Guibas, D. E. Knuth, and M. Sharir, "Randomized incremental construction of Delaunay and Voronoi diagrams," *Algorithmica*, vol. 7, no. 4, pp. 381–413, 1992.

[11] J. M. Augenbaum and C. S. Peskin, "On the construction of the Voronoi mesh on a sphere," *Journal of Computational Physics*, vol. 59, no. 2, pp. 177–192, 1985.

[12] R. J. Renka, "Algorithm 772: STRIPACK: Delaunay triangulation and Voronoi diagram on the surface of a sphere," *ACM Transactions on Mathematical Software*, vol. 23, no. 3, pp. 416–434, 1997.

[13] H.-S. Na, C.-N. Lee, and O. Cheong, "Voronoi diagrams on the sphere," *Computational Geometry: Theory and Applications*, vol. 23, no. 2, pp. 183–194, 2002.

[14] S. Fortune, "A sweepline algorithm for Voronoi diagrams," *Algorithmica*, vol. 2, pp. 153–174, 1987.

[15] F. Dehne and R. Klein, "A sweepcircle algorithm for Voronoi diagrams (extended abstract)," in *Graph-Theoretic Concepts in Computer Science: Proceedings of the International Workshop WG'87*, ser. Lecture Notes in Computer Science, H. Göttler and H. J. Schneider, Eds., vol. 314. Springer, 1988, pp. 59–69.

[16] L. J. Guibas and J. Stolfi, "Ruler, compass, and computer: The design and analysis of geometric algorithms," DEC/HP Labs, Systems Research Center, Palo Alto, CA, Tech. Rep. SRC-RR-37, Feb. 1989. [Online]. Available: http://www.hpl.hp.com/techreports/Compaq-DEC/SRC-RR-37.html

[17] M. T. Goodrich and R. Tamassia, *Data Structures and Algorithms in JAVA (fourth edition)*. Hoboken, NJ: Jonh Wiley & Sons, Inc., 2006.

[18] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms (second edition)*. Cambridge, MA: MIT Press, 2001.

[19] E. Høg, C. Fabricius, V. V. Makarov, S. Urban, T. Corbin, G. Wycoff, U. Bastian, P. Schwekendiek, and A. Wicenec, "The Tycho-2 catalogue of the 2.5 million brightest stars," *Astronomy and Astrophysics*, vol. 355, pp. 27–30, January 2000.

[20] K. L. Clarkson, "Hull (version 1.0)," 1995. [Online]. Available: http://netlib.org/voronoi/hull.html

[21] "CGAL: Computational Geometry Algorithms Library (version 3.6)," 2010. [Online]. Available: http://www.cgal.org

[22] K. L. Clarkson, K. Mehlhorn, and R. Seidel, "Four results on randomized incremental constructions," *Computational Geometry: Theory and Applications*, vol. 3, no. 4, pp. 185–212, 1993.

[23] C. B. Barber and H. Huhdanpaa, "Qhull (version 2010.1)," 2010. [Online]. Available: http://www.qhull.org

[24] J.-D. Boissonnat, O. Devillers, S. Pion, M. Teillaud, and M. Yvinec, "Triangulations in CGAL," *Computational Geometry: Theory and Applications*, vol. 22, no. 1-3, pp. 5–19, 2002.

[25] M. Caroli, P. M. M. De Castro, S. Loriot, O. Rouiller, M. Teillaud, and C. Wormser, "Robust and eficient Delaunay triangulations of points on or close to a sphere," INRIA, Research Report RR-7004, 2009.

[26] K. Mehlhorn and S. Näher, "LEDA: C++ Library of Efficient Data types and Algorithms," 1999. [Online]. Available: http://www.algorithmic-solutions.com/leda/

[27] K. L. Clarkson, "Safe and effective determinant evaluation," in *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science (FOCS'92)*. IEEE Computer Society, 1992, pp. 387–395.

[28] J.-D. Boissonnat, O. Devillers, and S. Hornus, "Incremental construction of the Delaunay triangulation and the Delaunay graph in medium dimension," in *Proceedings of the 25th annual symposium on Computational geometry (SCG'09)*. New York, NY, USA: ACM, 2009, pp. 208–216.