Output sensitive construction of the 3D Delaunay triangulation of constrained sets of points

Jean-Daniel Boissonnat, André Cerezo, Olivier Devillers, Monique Teillaud

Output sensitive construction of the 3D Delaunay triangulation of constrained sets of points*

Algorithme dépendant de la sortie pour la triangulation de Delaunay 3D d'ensembles de points contraints

Jean-Daniel Boissonnat[†] **André Cérézo**[‡] **Olivier Devillers**[†]
Monique Teillaud[†]

[†]INRIA, B.P.93, 06902 Sophia-Antipolis cedex (France),
[†]Phone : +33 93 65 77 62, E-mail : Firstname.Name@inria.fr
[‡]Université de Nice, Parc Valrose, 06034 Nice cedex (France)
This work has been supported in part by the ESPRIT Basic Research Action Nr. 3075 (ALCOM).

Abstract

In this paper, two algorithms are presented to compute the Delaunay triangulation of a set $\mathcal{S}$ of n points in 3-dimensional space when the points lie in a set $\mathcal{P}$ of k planes. If $k = 2$ the algorithm runs in time $O(n \log n + t)$ where t is the size of the output ; if $k \geq 3$ the time bound is $O(tk \log n)$. In both cases, the storage is $O(n)$. The second algorithm is extended to the case of points lying in k p-dimensional hyperplanes of $\mathbb{R}^d$.

Résumé

Cet article présente deux algorithmes pour le calcul de la triangulation de Delaunay de n points dans l'espace lorsque les points sont contraints à appartenir à k plans. Si $k = 2$ l'algorithme termine en temps $O(n \log n + t)$ où t est la taille du résultat ; si $k \geq 3$ la complexité est $O(tk \log n)$. Dans les deux cas, la mémoire nécessaire est $O(n)$. Le deuxième algorithme s'e'tend au cas de k sous-espaces de dimensions p de $\mathbb{R}^d$.

1 Introduction

The Delaunay triangulation and its dual, the Voronoï diagram, are fundamental structures in computational geometry.[PS85,Ede87,Aur91] In two dimensions, the size of these structures, for n points, is $O(n)$ and there exist several optimal $O(n \log n)$ algorithms for their construction. In higher dimensions, the size of the output depends on the input distribution and may vary from $O(n)$ to $O(n^{\lfloor \frac{d+1}{2} \rfloor})$ where d denotes the dimension. In particular, in 3-dimensional space, the Delaunay triangulation of a set $\mathcal{S}$ of n sites may consist of a number of tetrahedra which may vary from $O(n)$ to $O(n^2)$. Thus it is worth seeking algorithms whose complexities depend on both the input size n and the output size t.

As is well known, constructing the Delaunay triangulation (or the Voronoï diagram) of points in d-dimensional space can be reduced to constructing a convex hull in $(d+1)$-dimensional space. Thus the output sensitive algorithm of Seidel for constructing convex hulls in higher dimensions[Sei86] can be used to construct the Delaunay triangulation of n sites in an output sensitive manner. The time complexity of the original algorithm as described by Seidel is $O(n^2 + t \log n)$ for any $d \geq 3$. A very recent result of Matoušek mentioned in Theorem 5 allows to reduce the complexity to $O(n^{4/3} + t \log n)$; with Matoušek modification, this algorithm is rather complicated and still not satisfying if t is $o(n^{4/3})$.

Other algorithms are known that are sensible to the output for some specific types of inputs. Incremental randomized algorithms, for instance, are sensitive to the maximal size of the successive triangulations.[CS89,BDS*] Though their time complexity may be output sensitive in some cases (in particular, these algorithms are linear if the input sites are uniformly distributed), it may well happen that an intermediate triangulation is much bigger than the final one. So these algorithms are not sensitive to the output in bad cases.

A deterministic algorithm whose complexity depends on the output (and in fact reaches the optimal time bound $O(n \log n + t)$) is known in the special case of points in 3-dimensional space constrained to lie in two parallel planes.[Boi88]

This paper extends on this last result and presents two algorithms to compute the Delaunay triangulation of a set $\mathcal{S}$ of n points in 3-dimensional space when the points lie in a set $\mathcal{P}$ of k planes. The algorithm for the case of two (not necessarily parallel) planes runs in optimal time $O(n \log n + t)$. If $k \geq 3$ the time complexity is $O(tk \log n)$. In both cases, the storage is $O(n)$.

This last algorithm compares favorably with Seidel's one[Sei86] when the number of planes is small with respect to the number of points. This is in particular true when dealing with tomographic images and 3-dimensional shape reconstruction problems.[Boi88] Our algorithms are quite easy to code and appear to be good candidates for practical implementations.

Using Matoušek's recent result, this algorithm can be extended to sets of points of $I\!R^d$ lying in k p-dimensional hyperplanes.

We now briefly outline the algorithm in 3-dimensional space : the triangulation is constructed incrementally. Consider a tetrahedron $abcd$ of the triangulation, the basic

problem is to find the tetrahedra that share faces with it. This is equivalent to finding for each face of $abcd$, say abc, the empty sphere that passes through abc and another site $e(\neq d)$. The searched sphere, if it exists, is in the pencil of spheres that passes through abc. The key is to avoid testing a linear number of points in order to find e. This can be done by using the fact that the sites lie in k planes. In particular, observe that the intersection of the pencil of spheres with a plane P is a pencil of circles in P. We seek a site s_P in each plane P that has a member of the pencil of circles passing through it such that the circle does not contain any site in its interior. The site e is the site $s_P \neq d$ such that the sphere circumscribing $abcs_P$ is empty.

The key to find quickly s_P is described in Section 2 and the whole algorithm is given in Section 3. Section 4 deals with the particular case of only two planes.

2 Preliminary results

As mentioned in the introduction, pencils of circles play a central role in our algorithm. this section provides the necessary material about pencils of circles and solves a query problem which will be a basic ingredient of our algorithm.

2.1 Space of circles and pencils of circles

We take interest in the set of circles drawn in plane P. This set of circles is represented by a three dimensional space $\mathcal{C}_P$: the space of circles. Let $x^2 + y^2 - 2\varphi x - 2\psi y + \chi = 0$ be the equation of a circle in P ; this circle is represented by the point (φ, ψ, χ) in $\mathcal{C}_P$.

In this section, we derive the properties of $\mathcal{C}_P$ from well known results on polarity. Necessary notions and standard vocabulary (pencil, radical axis, polar plane, conjugate...) are defined in Coxeter or Pedoes's books[Cox61,Ped70] for example.

- The set of circles of radius 0 is a paraboloid of equation : $\varphi^2 + \psi^2 = \chi$. A point in P is often identified with the 0 radius circle centered at this point. If plane P is identified to the plane $\chi = 0$ then the 0 radius circle is obtained by raising the point on the paraboloid. The interior of the paraboloid is the set of "imaginary" circles (with negative square radius) (see Figure 1). Notice that the horizontal projection of a point in $\mathcal{C}_P$ is the center of the circle, and the χ-coordinate is the power of the origin with respect to the circle.

- Two circles $(\varphi_1, \psi_1, \chi_1)$ and $(\varphi_2, \psi_2, \chi_2)$ are orthogonal if the corresponding points are conjugate with respect to the paraboloid or equivalently if $\chi_1 + \chi_2 = 2\varphi_1\varphi_2 + 2\psi_1\psi_2$.

- As a particular case, the set of circles passing through $M_0 \in P$ is also the set of circles perpendicular to the 0 radius circle M_0, which is in $\mathcal{C}_P$ the polar plane of M_0. But, as M_0 belongs to the paraboloid, the polar plane is nothing else than the tangent plane to the paraboloid at M_0. For a circle in the half space (limited by this plane) which does not contain the paraboloid, M_0 is inside the circle. For a circle in the other half space, M_0 is outside.

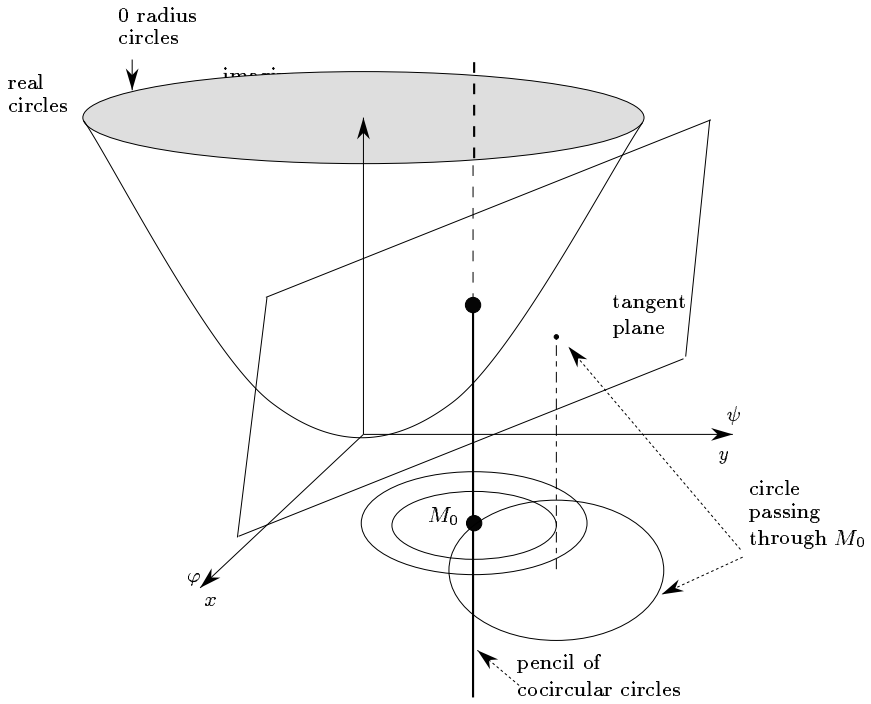

Figure 1: The space of circles

- A pencil of circles, that is the set of linear combinations of two circles, transforms in $\mathcal{C}_P$ into the line through the two points. A pencil of circles with limiting points is a line hitting the paraboloid in the two limiting points (see Figure 2). A pencil of circles with common points is a line which does not hit the paraboloid, this line is the intersection of the two planes tangent to the paraboloid at the common points (see Figure 3). A concentric pencil is a line parallel to the χ axis (see Figure 1).

- The points at infinity of the projective closure of $\mathcal{C}_P$ correspond to the straight lines in P, namely the point at infinity in the direction of (φ, ψ, χ) is the straight line of equation $-2\varphi x - 2\psi y + \chi = 0$, and the point at infinity of a line in $\mathcal{C}_P$ is the radical axis of the corresponding pencil.

  Thus the pencils whose radical axis is a given line in P form in $\mathcal{C}_P$ the set of lines parallel to a given direction (orthogonal to the radical axis).

  In particular, a pencil of circles having the x-axis as its radical axis corresponds in $\mathcal{C}_P$ to a line parallel to the ψ-axis.

2.2 Empty circles and Voronoï diagrams

The properties of $\mathcal{C}_P$ above yield another interpretation of the well known correspondance between arrangements of planes and Voronoï diagrams.[ES86] This will be detailed below and used in the subsequent section. As noticed in the preceding section,

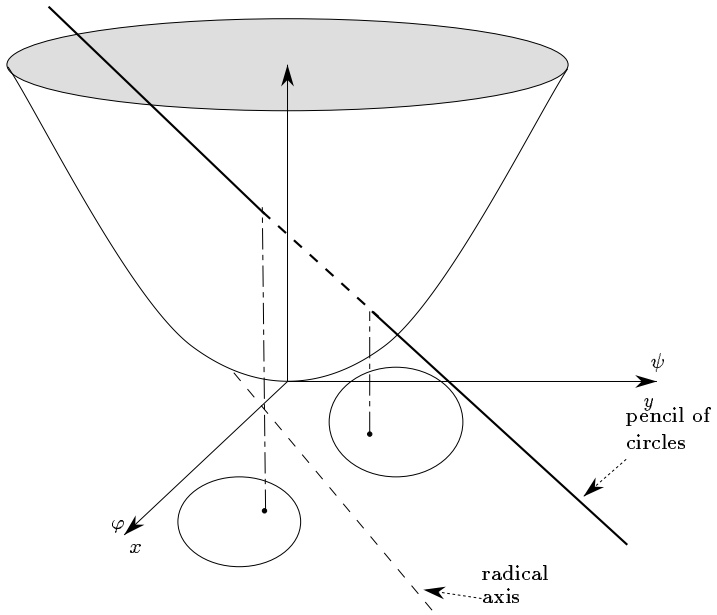

Figure 2: A pencil of circles with limiting points

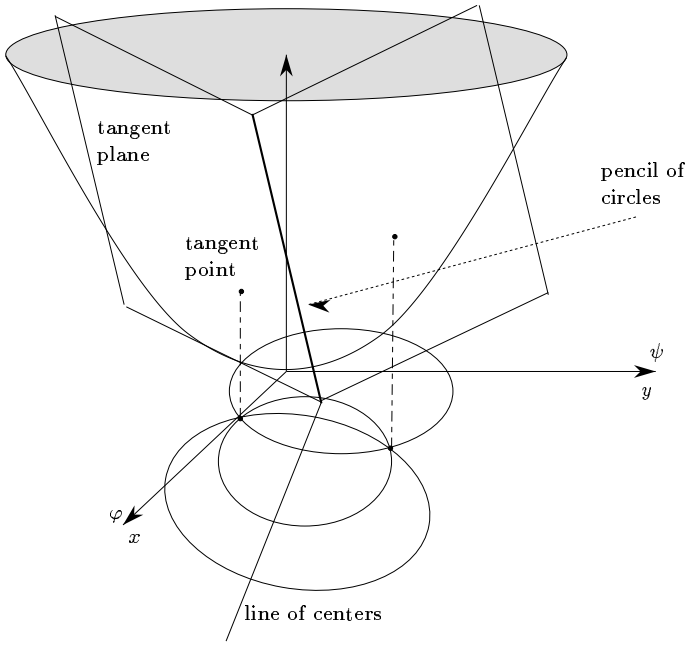

Figure 3: A pencil of circles with common points

the set of circles which do not contain a given point maps in $\mathcal{C}_P$ into an half space, limited by the polar plane of the point. Thus, if $\mathcal{S}_P$ is a set of sites in plane P, then the set of empty circles (i.e. the set of circles which do not surround any site of $\mathcal{S}_P$) of plane P is in $\mathcal{C}_P$ the intersection of the corresponding half spaces. It is a convex polyhedron $U_{\mathcal{S}_P}$, whose facets are tangent to the paraboloid. The intersection of this convex set with a line, representing a pencil of circles is made up of the extremal empty circles of this pencil (see Figure 4). In general there are zero or two such extremal circles. These circles are the empty circles of the pencil passing through a point of $\mathcal{S}_P$. Because there is exactly one circle of a pencil passing through a given point, the determination of the extremal empty circles of a given pencil can be viewed equivalently, as the determination of the points of $\mathcal{S}_P$ on these circles.

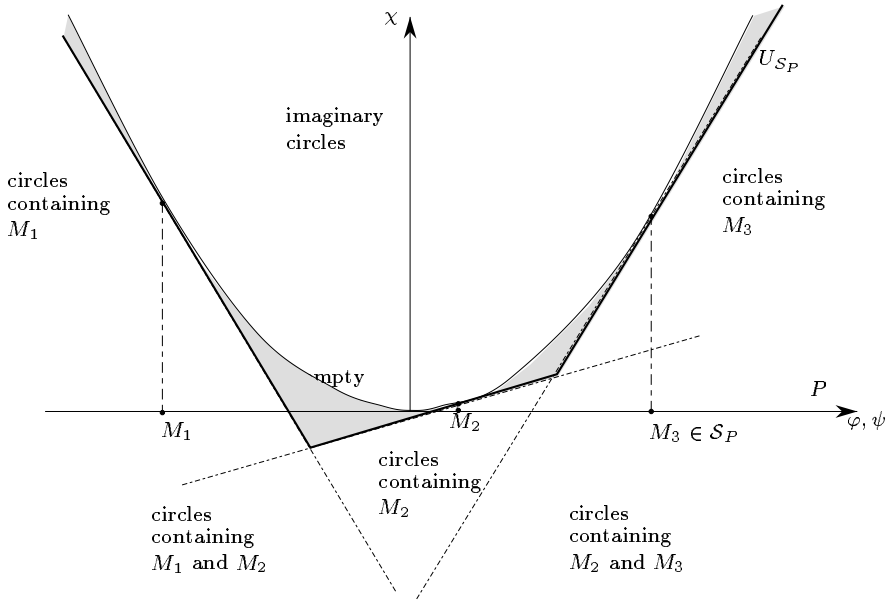

Figure 4: Set of empty circles

In particular, for a concentric pencil, the intersection with a vertical line in $\mathcal{C}_P$ gives the largest empty circle having a given center. This circle passes through a site of $\mathcal{S}_P$, namely the site of $\mathcal{S}_P$ which is the closest to the center. In other words the projection of $U_{\mathcal{S}_P}$ on the horizontal plane is the Voronoï diagram of $\mathcal{S}_P$.

2.3 A query problem

Lemma 1 *Let $\mathcal{S}_P$ be a set of sites in plane P, and F be a pencil of circles, the extremal empty circles of F can be found in $O(\log n)$ time using $O(n)$ space and $O(n \log n)$ preprocessing time.*

Proof. The algorithm first constructs $U_{\mathcal{S}_P}$ and then computes in $\mathcal{C}_P$ the intersection of the line representing F with the convex $U_{\mathcal{S}_P}$, which can be done within the given

bounds.[EM85] If there is no intersection between the line and $U_{\mathcal{S}_P}$, then all the circles of the pencil contain some points of $\mathcal{S}_P$. □

3 Delaunay triangulation of n points in k planes

This section deals with the main problem of this paper : the construction of the Delaunay triangulation of a set $\mathcal{S}$ of n points belonging to a set $\mathcal{P}$ of k planes. We suppose that no five points are cospherical, which implies that in each plane four points cannot be cocircular. This hypothesis of non degenerate positions can be removed using standard perturbations techniques.[EM90]

The algorithm starts from a first tetrahedron and enumerates all the Delaunay tetrahedra in a suitable order to be defined later. Lemma 1 is used as described in Section 3.1 as a hint to determine the Delaunay neighbor of an already constructed tetrahedron through one of its facets.

3.1 Searching for a neighbor

Lemma 1, which solves a 2-dimensional problem, can be used to solve 3-dimensional queries in a special case. Let P be a plane in the Euclidean 3-dimensional space, and Q be the radical plane of a pencil of spheres (the radical plane is the sphere of the pencil with infinite radius) ; then the intersection of the spheres with P is a pencil of circles whose radical axis is $P \cap Q$ (see Figure 5). Let $\mathcal{S}_P$ be a set of sites in plane P. The

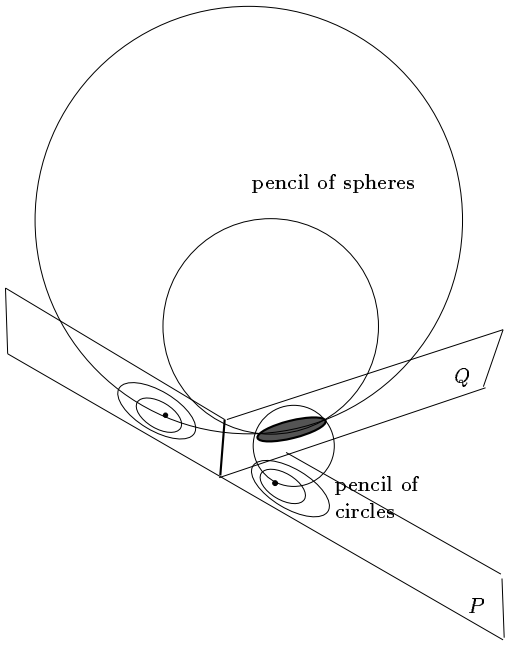

Figure 5: Intersection of a pencil of spheres with a plane.

extremal empty spheres (if there exist any) which do not contain any point of $\mathcal{S}_P$ are obviously determined by the extremal empty circles of the pencil of circles. Thanks to Lemma 1, this can be done in $O(\log n)$ time.

If all the planes P in $\mathcal{P}$ are considered now, then the extremal empty spheres which do not contain any point of $\mathcal{S} = \cup_{P \in \mathcal{P}} \mathcal{S}_P$ can be determined in two steps. Firstly, for each plane P, the extremal spheres which do not contain any point of $\mathcal{S}_P$ are determined. Then, we select from the at most $2k$ candidate spheres the possible solutions as follows.

Let $\Sigma_1, \Sigma_2, \ldots, \Sigma_p$ $p \leq 2k$ be the candidate spheres. Each Σ_i passes through a point p_i of $\mathcal{S}$. We examine in turn each candidate sphere and, at step i, $i = 1 \ldots p$, we maintain the (at most two) spheres which do not contain any $p_j, j \leq i$. This can easily be done in $O(k)$ time.

In this way, it is possible to find the neighbors of a given Delaunay tetrahedron. Let *abc* be a face of the Delaunay triangulation, this face belongs in general to two Delaunay tetrahedra *abcd* and *abce*. The circumscribing spheres of these two tetrahedra are the two extremal empty spheres of the pencil of spheres with common points *abc*.

Lemma 2 *A set $\mathcal{S}$ of n points lying in k planes can be preprocessed in $O(n \log n)$ time and $O(n)$ space, so that the two Delaunay tetrahedra adjacent to a given Delaunay triangle can be determined in $O(k \log n)$ time.*

3.2 The main algorithm

These remarks yield a simple algorithm to construct the Delaunay triangulation of a set of n points lying in k planes. The algorithm starts from one initial tetrahedron and then constructs the whole set of Delaunay tetrahedra incrementally. In order to achieve a linear space complexity, we process the tetrahedra in an order that ensures that the union of already constructed tetrahedra remains simply connected at each step of the incremental construction.

This can be ascertained by making use of the mechanism of shelling introduced by Bruggesser and Mani[BM71] for proving that the boundary complex of any convex polytope is shellable in any dimension[BM71, Proposition 2]. This idea has already been exploited by Seidel[Sei86] and the reader can also refer to his paper.

The 3-dimensional Delaunay triangulation is interpreted as a 4-dimensional simplicial lower convex hull $\mathcal{L}$ of points on a paraboloid. Let D be a line parallel to the axis of the paraboloid, and imagine an observer moving down along D from the intersection point of D and $\mathcal{L}$. This observer reports first the facet F_1 of $\mathcal{L}$ hit by D and, as he moves down along D, he reports the facets $F_2, \ldots, F_t$ in the order they become visible (see Figure 6). This order called the shelling order is given by the attribution of a priority to each facet of $\mathcal{L}$. The priority of a facet is the altitude of the intersection of its supporting hyperplane with D and the shelling order of the facets corresponds to the decreasing order of these priorities. Bruggesser and Mani[BM71] ensures that this order is a good shelling of every convex polytope, i.e. that for all i, $\cup_{j=1}^{i} F_j$ is a simply connected 3-dimensional topological ball in 4-dimensional space.

Going back to the original problem of constructing 3-dimensional Delaunay triangulation, we are able to compute the priority of each tetrahedron ; and, if the tetrahedra

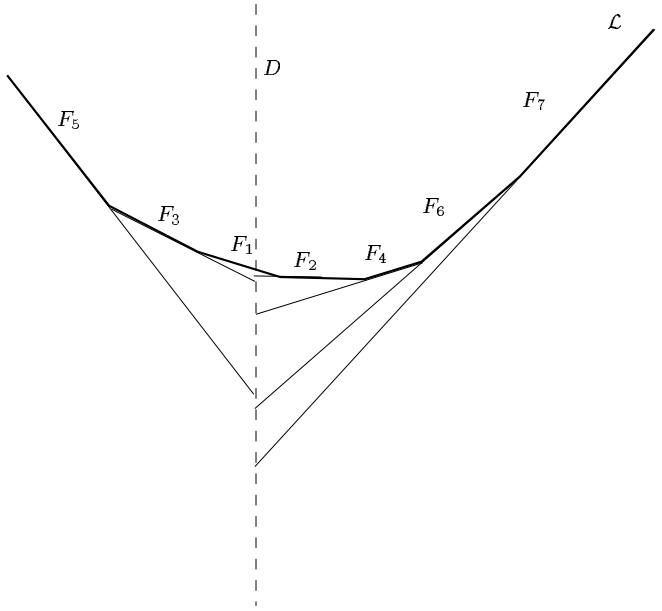

Figure 6: The shelling order

are constructed by decreasing priority, the set of constructed tetrahedra remains simply connected at each step. We assume that line D is in general position, and there are no two distinct tetrahedra with the same priority (degeneracies can be solved by perturbing D, see the next section).

Let $\mathcal{T}$ be the set of already constructed tetrahedra. We maintain in a priority queue the set of tetrahedra not in $\mathcal{T}$, but sharing a face with the tetrahedra of $\mathcal{T}$. Notice that, by the result of Bruggesser and Mani[BM71] a tetrahedron of the queue shares one, two or three faces with $\mathcal{T}$; thus it appears one two or three times in the priority queue (with the same priority). A tetrahedron is present only once if and only if one of its vertices has never been considered so far.

As all the Delaunay tetrahedra adjacent to $\mathcal{T}$ are present in the priority queue, we can ensure that the next tetrahedron in the shelling is the first element of the queue (i.e., the one with the highest priority).

Thus our algorithm will run as follows :

1. Execute the preprocessing step of Lemma 2.
2. Initialization :
   (a) Compute a triangle $a_1b_1c_1$ of the convex hull (for example the first face produced by the gift wrapping algorithm[PS85]).
   (b) Compute $d_1 = p_{a_1b_1c_1}$ using Lemma 2.
   (c) Take as D any "vertical" line in $\mathbb{R}^4$ passing through a point p_D of the interior of the facet $a_1b_1c_1d_1$ lifted on the 4-dimensional paraboloid (notice that, by

construction of D, $a_1b_1c_1d_1$ is the first face in the shelling order induced by D).

(d) Insert $a_1b_1c_1d_1$ in the priority queue.

3. Repeat

   (a) Find the first tetrahedron $abcd$ in the priority queue,

   (b) If $abcd$ appears only once, without loss of generality $d = p_{abc}$. Insert tetrahedron $abcd$ in $\mathcal{T}$, compute p_{abd}, p_{acd} and p_{bcd}, insert $abdp_{abd}$, $acdp_{acd}$ and $bcdp_{bcd}$ in the priority queue.

   (c) If $abcd$ appears twice, without loss of generality $d = p_{abc}$ and $c = p_{abd}$. Insert tetrahedron $abcd$ in $\mathcal{T}$, compute p_{acd} and p_{bcd}, insert $acdp_{acd}$ and $bcdp_{bcd}$ in the priority queue.

   (d) If $abcd$ appears three times, without loss of generality $d = p_{abc}$, $c = p_{abd}$ and $b = p_{acd}$. Insert tetrahedron $abcd$ in $\mathcal{T}$, compute p_{bcd}, insert $bcdp_{bcd}$ in the priority queue.

   Until the queue is empty.

3.3 Dealing with degeneracies

In this section, we explain how to deal with degeneracies that may occur due to an unlucky choice of D. D is a vertical line, in 4-dimensional space, determined by the first third coordinates of p_D : x_D, y_D and z_D. D is used to sort some hyperplanes according to their intersection points with D.

If D passes through some vertices, edges or 2-faces of the arrangement of hyperplanes, then we are in a degenerate case : D induces only a partial order on the hyperplanes. But, it is clear, for continuity reasons, that it is possible to translate D by a small horizontal vector $\vec{\varepsilon}(x_\varepsilon, y_\varepsilon, z_\varepsilon)$ such that the order induced by $D+\vec{\varepsilon}$ is a total order compatible with the order induced by D.

The algorithmic way to find $\vec{\varepsilon}$ is the following :

- If D passes through some vertex v of the arrangement of hyperplanes, we translate D to D_1 in the x direction by a sufficiently small amount so that no 2-face of the arrangement is crossed.

  The 2-faces incident to v cannot be crossed by D during its move. However, if such a face f is parallel to the x direction, all the vertical lines between D and D_1 will intersect f. The situation may still be degenerate, if D has been translated along an edge incident to v. Notice however that, for a sufficiently small perturbation, it is now impossible that D passes through a vertex of the arrangement. In such a case we translate D as indicated below.

- If D passes through some edge e (but not a vertex) of the arrangement of hyperplanes, we translate D_1 to D_2 in the y direction by a sufficiently small amount so that no 2-face of the arrangement is crossed.

For a sufficiently small perturbation, it is now impossible that D passes through an edge of the arrangement, but still possible that D intersect a 2-face, when one of the 2-faces containing e is parallel to the y-axis. So we perform, one more perturbation.

- If D passes through some 2-face of the arrangement of hyperplanes, we translate D_2 to $D_3 = D + \vec{\varepsilon}$ in the z direction by a sufficiently small amount so that no 2-face of the arrangement is crossed.

We do not need to compute the exact value of $\vec{\varepsilon}$. In fact the line D is perturbated only when a degeneracy is encountered, this degeneracy is solved locally, i.e. only the order on the hyperplanes involved in the current degeneracy is computed. Once the degeneracy is solved, we use again the initial line D.

More precisely, if several points p_{abc} are on the top of the priority queue Q with the same priority, we construct an auxiliary priority queue Q_1 where the priorities are computed according to a line D_1 (D translated in the x direction). The only points inserted in Q_1 are those with the same (and highest) priority in Q ; in that way, the amount of the x translation does not matter since the only 2-faces in the 4-dimensional arrangement of hyperplanes present in Q_1 are all adjacent to v and thus cannot be crossed when D moves to D_1. When Q_1 is empty, we can go back to the use of Q. If some points have the same priority in Q_1, we create an auxiliary queue Q_2 to solve this degeneracy, and possibly a queue Q_3. From the reasons above, four queues are sufficient.

3.4 Complexity

The complexity of this algorithm can be analyzed as follows.

Step 1 is completed in $O(n \log n)$ time and uses $O(n)$ space by Lemma 2.

Step 2a is done in $O(n)$ time.

Step 2b takes $O(k \log n)$ time by Lemma 2.

Step 2c and 2d are done in constant time.

At each iteration of Step 3 a new tetrahedron is added to the Delaunay triangulation thus this step is executed t times. Operations on the priority queue are done in $O(\log n)$ time (even if auxiliary priority queues are used to solve degeneracies), the determination of a point p_{xyz} is done in $O(k \log n)$ time by Lemma 2 and the computation of the priority of a tetrahedron takes constant time. Thus the overall cost of Step 3 is $O(tk \log n)$.

Each item in the priority queue corresponds to a triangle on the boundary of the already constructed tetrahedra. As the tetrahedra are reported in a shelling order, these triangles form a topological sphere, and by Euler's relation there is at most a linear number of such triangles. So the size of the priority queue is $O(n)$.

Theorem 3 *The three dimensional Delaunay triangulation of n points lying in k planes can be computed using $O(tk \log n)$ time and $O(n)$ extra space where t is the size of the output.*

4 Case of two planes

In the special case of only two planes, it is possible to speed up the algorithm. This is in fact a generalization of the algorithm for two parallel planes P and Q.[Boi88]

In Boissonnat's paper[Boi88], it was shown that the three dimensional Voronoï diagram (and thus the Delaunay triangulation) of $\mathcal{S}_P \cup \mathcal{S}_Q$ can be obtained by superimposing the two 2-dimensional Voronoï diagrams of $\mathcal{S}_P$ and $\mathcal{S}_Q$. It will be shown in the appendix that the situation for non parallel planes is quite similar : the 3-dimensional Delaunay triangulation is obtained by superimposing the Voronoï diagrams of $\mathcal{S}_P$ and $\mathcal{S}_Q$ for the hyperbolic metric (recalled in the appendix). However, using the space of circles, it is possible to derive more tractable (linear) diagrams that can be used instead of the hyperbolic ones.

Let P and Q be the two planes and denote l the common line $P \cap Q$. As in the preceding section, the choice of a circle c in P determines a pencil of spheres : the set of spheres intersecting P along c. The intersection of this pencil with Q is a pencil of circles in Q. So, to a point c of $\mathcal{C}_P$ we can associate a line $\delta(c)$ in $\mathcal{C}_Q$. Using a judicious choice of coordinates in P and Q this transformation is trivial. More precisely, by choosing in the two systems of coordinates of the planes, the origin on l and l as the x-axis, the pencil of circles in $\mathcal{C}_Q$ determined by the point c of coordinates (φ, ψ, χ) in $\mathcal{C}_P$ is exactly the line $\delta(c)$ parallel to the ψ-axis passing through the point of coordinates (φ, ψ, χ) in $\mathcal{C}_Q$, since l is the radical axis of the pencil (see Section 2.1 and Figure 7).

In the sequel, $\mathcal{C}_P$ and $\mathcal{C}_Q$ will be identified and denoted $\mathcal{C}$.

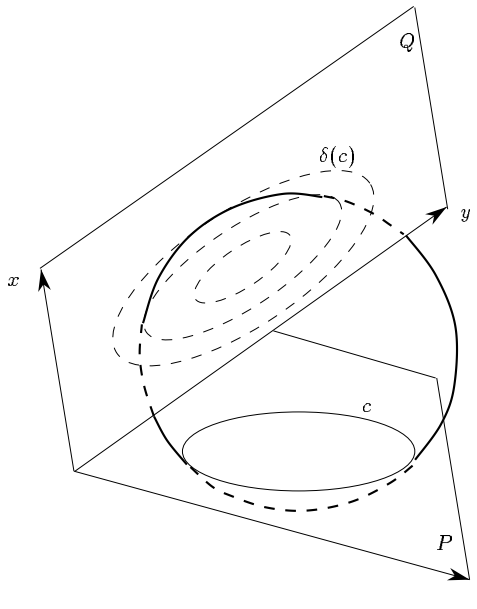

Figure 7: Case of two planes

Suppose now that *abcd* is a Delaunay tetrahedron. Two cases may occur : three of the vertices are in the same plane and one in the other, or there are two vertices in

each plane.

In the first case, let a, b and c be the vertices in P and d be the fourth vertex in Q. The intersection of the sphere circumscribing $abcd$ with P is the circle circumscribing abc represented in $\mathcal{C}$ by a vertex v of $U_{\mathcal{S}_P}$. The intersection of the sphere with Q is represented in $\mathcal{C}$ by a point that must lie on the line parallel to the ψ-axis and passing through v. This line hits $U_{\mathcal{S}_Q}$ in at most two points representing the possible empty extremal circles in Q. The (at most) two faces of $U_{\mathcal{S}_Q}$ hit by the line correspond to the two possible points d.

In the second case, let $a, b \in P$ and $c, d \in Q$. For a similar reason, the circles which are the intersection of the sphere with P and Q lie on an edge of $U_{\mathcal{S}_P}$ and $U_{\mathcal{S}_Q}$ respectively. These two points of $\mathcal{C}$ are on the same line parallel to the ψ-axis.

Since all relevant lines are parallel to the ψ-axis, the problem can be studied in projection on a plane Π of $\mathcal{C}$ perpendicular to the ψ-axis.

At this time, it is possible to describe the algorithm. $U_{\mathcal{S}_P}$ (*resp.* $U_{\mathcal{S}_Q}$) is computed in the coordinate system described above and is projected onto Π. We distinguish the front projection $U^f_{\mathcal{S}_P}$ (*resp.* $U^f_{\mathcal{S}_Q}$) i.e. the projection of the faces of $U_{\mathcal{S}_P}$ (*resp.* $U_{\mathcal{S}_Q}$) visible from the point $(0, +\infty, 0)$; and the back projection $U^b_{\mathcal{S}_P}$ (*resp.* $U^b_{\mathcal{S}_Q}$) (as seen from $(0, -\infty, 0)$).

We can now restate the above facts. For each vertex v of $U^f_{\mathcal{S}_P}$ or $U^b_{\mathcal{S}_P}$ (*resp.* $U^f_{\mathcal{S}_Q}$ or $U^b_{\mathcal{S}_Q}$) (which corresponds to a Delaunay circle circumscribing some triangle abc in P (*resp.* Q)) there are (at most) two possible Delaunay tetrahedra. The two points in Q (*resp.* P) to complete the triangle abc are represented in Π by the faces of $U^f_{\mathcal{S}_Q}$ and $U^b_{\mathcal{S}_Q}$ (*resp.* $U^f_{\mathcal{S}_P}$ or $U^b_{\mathcal{S}_P}$) containing this vertex.

To each intersection of an edge e of $U^f_{\mathcal{S}_P}$ or $U^b_{\mathcal{S}_P}$ (representing an empty circle in P passing through points ab) with an edge e' of $U^f_{\mathcal{S}_Q}$ or $U^b_{\mathcal{S}_Q}$(representing an empty circle in Q passing through points cd) corresponds the Delaunay tetrahedron $abcd$.

Thus the algorithm consists in superimposing the convex planar subdivisions and in constructing the resulting subdivisions.

- $U^f_{\mathcal{S}_P}$ and $U^f_{\mathcal{S}_Q}$,
- $U^b_{\mathcal{S}_P}$ and $U^f_{\mathcal{S}_Q}$,
- $U^f_{\mathcal{S}_P}$ and $U^b_{\mathcal{S}_Q}$,
- $U^b_{\mathcal{S}_P}$ and $U^b_{\mathcal{S}_Q}$.

Each of these superimpositions corresponds to the triangulation of one of the four wedges defined by the two planes. These superimpositions can be computed in time $O(n+t)$ where n is the number of points and t the number of tetrahedra if one uses the algorithm of Guibas and Seidel[GS86] based on the idea of a topological sweep.[EG86] This complexity is optimal with respect to the input and the output sizes.

Theorem 4 *The three dimensional Delaunay triangulation of n points lying in 2 planes can be computed in $O(t + n \log n)$ time and $O(n)$ space where t is the size of the output.*

It is shown in the appendix that computing $U^f_{\mathcal{S}_P}$ (and similarly for the others) is in fact equivalent to computing the hyperbolic Voronoï diagram of the sites in one of the halves of P limited by l. When P is parallel to Q the preceeding discussion can be easily adapted : pencils of circles are now concentric pencils and only one "wedge" remains relevant (the slice between P and Q).

5 Higher dimensions

The first algorithm can be adapted in a straightforward way to the case of points belonging to k 2-dimensional planes of $\mathbb{R}^d$. Extension to k 1-dimensional lines is also easy. The general outline of the algorithm and its time complexity are not changed, i.e. $kt \log n$.

The case of points of $\mathbb{R}^d$ belonging to k p-dimensional hyperplanes ($p > 2$) is more difficult since it requires to perform intersection queries between a line and a convex $(p+1)$-dimensional polytope.

We present a solution to that problem, based on the following recent result by Matoušek [Mat91].

Theorem 5 [Matoušek] *One can preprocess a polytope defined as the intersection of n half spaces of $\mathbb{R}^\delta$ in $O(n \log n)$ (deterministic) time and using $O(n)$ space so that, given a query line, the intersection points between the line and the polytope can be computed in $O\left(n^{1-\frac{1}{\lfloor \frac{\delta}{2} \rfloor}}\right)$ time.*

If m is a parameter, $n \leq m \leq n^{\lfloor \frac{\delta}{2} \rfloor}$, and if space and preprocessing $O(m^{1+\epsilon})$ for some fixed ϵ are allowed, the query can be answered in time $O\left(\frac{n}{m^{1/\lfloor \frac{\delta}{2} \rfloor}} (\log n)^{O(1)}\right)$.

The idea of our algorithm is to start with little memory and run the algorithm until the cumulated time of all the queries performed so far exceeds the preprocessing time and the memory storage. At that point, we can allow more memory. Thus we restart a new preprocessing with a bigger value of m. And repeat that process (each such process will be called a step) until all the simplices have been constructed.

Let $T(i)$ be the time spent in step i, m_i be the parameter chosen at that step for the preprocessing and t_i be the number of simplices constructed at that step. We have ($\delta = p + 1$) :

$$T(i) = O(m_i^{1+\epsilon}) + O\left(\frac{n t_i (\log n)^{O(1)}}{m_i^{1/\lfloor \frac{\delta}{2} \rfloor}}\right)$$

We take m_i so as to balance the cost of the preprocessing and the cost of the queries performed at step i :

$$m_i = \left(nt_i(\log n)^{O(1)}\right)^{\frac{1}{\epsilon+1/\gamma}}$$

where

$$\gamma = \frac{1}{1+\frac{1}{\lfloor\frac{\delta}{2}\rfloor}}$$

In order to fulfill the hypotheses of Theorem 5, it is necessary that $m_i \leq n^{\lfloor\frac{\delta}{2}\rfloor}$; thus, if, at step i, $\left(nt_i(\log n)^{O(1)}\right)^{\frac{1}{\epsilon+1/\gamma}} > n^{\lfloor\frac{\delta}{2}\rfloor}$ we choose $m_i = n^{\lfloor\frac{\delta}{2}\rfloor}$ and we run the algorithm till the end of the construction without restarting the preprocessing step. We first suppose that this case does not happen. It follows from the expression of $T(i)$

$$\begin{aligned} T(i) &= O\left(\left(nt_i(\log n)^{O(1)}\right)^{\gamma\frac{1+\epsilon}{1+\gamma\epsilon}}\right) \\ &= O\left(\left(nt_i(\log n)^{O(1)}\right)^{\gamma(1+\epsilon)}\right) \end{aligned}$$

By taking $t_i = n^{1+\alpha i}$ for some $\alpha > 0$, and summing over the h steps needed to compute all the simplices, we obtain the overall computing time T of the algorithm :

$$T = O\left(\left(n^2(\log n)^{O(1)}\right)^{\gamma(1+\epsilon)} \sum_{i=1}^{h} n^{\alpha\gamma(1+\epsilon)i}\right)$$

The number of steps h is given by :

$$\sum_{i=1}^{h} t_i = t = n^{1+\alpha}\frac{n^{h\alpha}-1}{n^{\alpha}-1} \qquad (\star)$$

which implies

$$n^{h\alpha} = O(\frac{t}{n}) \text{ and } h = O(1)$$

We deduce that :

$$T = O\left(n^{\gamma+\epsilon}t^{\gamma+\epsilon}(\log n)^{O(1)}\right)$$

The algorithm above runs as described if m_i remains smaller than $n^{\lfloor\frac{\delta}{2}\rfloor}$, that is

$$\begin{aligned} m_h &\leq n^{\lfloor\frac{\delta}{2}\rfloor} \\ \left(nt_h(\log n)^{O(1)}\right)^{\frac{1}{\epsilon+1/\gamma}} &\leq n^{\lfloor\frac{\delta}{2}\rfloor} \\ nt_h &\leq n^{\lfloor\frac{\delta}{2}\rfloor(\epsilon+1+\frac{1}{\lfloor\frac{\delta}{2}\rfloor})} \\ t_h &\leq n^{\lfloor\frac{\delta}{2}\rfloor(1+\epsilon)} \end{aligned}$$

Furthermore it follows from $(\star)$ that $t \leq \frac{t_h}{1-n^{-\alpha}} \leq 2t_h$ for big enough values of n.

If this condition is not verified, the last step takes $O(\lfloor\frac{\delta}{2}\rfloor + \epsilon)$ preprocessing and performs t polylogarithmic queries.

Applying these results to our initial problem, we obtain

Theorem 6 *Constructing the Delaunay triangulation of n points constrained to belong to k p-dimensional ($p > 2$) hyperplanes of some Euclidean d-dimensional space can be done in time and space*

$$T = O\left(kn^{\gamma+\varepsilon}t'^{\gamma+\varepsilon} + kt''(\log n)^{O(1)}\right)$$

where ε is any positive constant,
t is the size of the output, $t' = \min(t, n^{\lfloor\frac{p+1}{2}\rfloor})$ and $t'' = t - t'$

$$\text{and} \quad \gamma = \frac{1}{1 + \frac{1}{\lfloor\frac{p+1}{2}\rfloor}} \qquad \left(\gamma = \frac{2}{3} \text{ for } p = 3, 4;\ \ \gamma = \frac{3}{4} \text{ for } p = 5, 6 \ldots\right)$$

6 Conclusion

We have presented output sensitive algorithms for constructing the Delaunay triangulation of special sets of points in 3-dimensional space. Specifically, when the input consists of n points scattered through k planes, we have shown that their Delaunay triangulation can be computed in $O(tk \log n)$ time and $O(n)$ space. This time bound can be reduced to $O(n \log n + t)$ (which is optimal with respect to the input and the output) when $k = 2$.

This result is a further step (after the case of two parallel planes[Boi88]) towards the efficient construction of the Delaunay triangulation in an output sensitive fashion in 3-dimensional space. We believe that these results are of interest in several practical applications, especially computer vision and computerized tomography where data are naturally distributed in planes. These results have been extended to higher dimensions, if the points belong to k p-dimensional subspaces of d-dimensional space. We let as an open question whether our construction can be improved with respect to k ; and we recall the main (and probably difficult) unsolved question in that area : *is $O(n \log n + t)$ an upper bound for constructing the Delaunay triangulation of n points in d-dimensional space if the triangulation consists of t tetrahedra ?*

Appendix. Hyperbolic Voronoï diagrams

This section presents an application of the space of circles in the context of (non Euclidean) hyperbolic geometry (a good introduction can be found in Beardon's book[Bea83]). The most classical conformal model of the 2-dimensional non Euclidean space (of Lobachevsky) is the Poincaré upper half plane $\mathbb{P} = \{x + iy \in \mathbb{C} \,|\, y > 0\}$ with the local metric $ds^2 = \frac{dx^2+dy^2}{y^2}$; the geodesics are the half-circles in $\mathbb{P}$ centered on the real axis (exceptionnally the vertical straight half-lines), and the (so called "hyperbolic") distance between two points $z, z' \in \mathbb{P}$ is, up to a constant factor, the logarithm of the (real) cross-ratio $(z, z', a, a') = \frac{z-a}{z-a'} \cdot \frac{z'-a'}{z'-a}$ where a, a' are the traces on the real axis of the geodesic through zz'.

The bisecting line δ of z and z' is the upper half of the circle Δ centered on the real axis and belonging to the pencil with limiting points z and z' (See Figure 8).

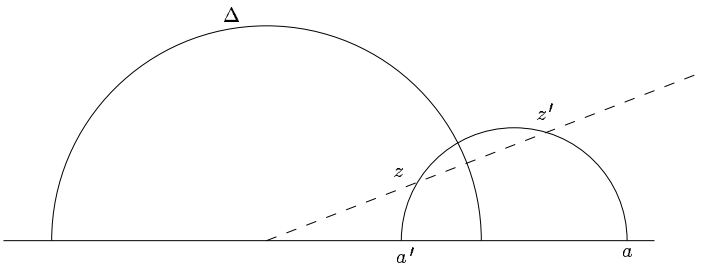

Figure 8: The bisecting line Δ of z and z'.

Thus one can define in the usual way the hyperbolic Voronoï diagram of a finite set $\mathcal{S}$ of sites in $I\!P$. The cell of $z \in \mathcal{S}$ is the set of points nearer (in the hyperbolic sense) to z than to any other points in $\mathcal{S}$. This is a (hyperbolically) convex subset of $I\!P$, limited by arcs of bissecting lines. The vertices are the centers (in the hyperbolic sense) of circles (in both hyperbolic an Euclidean sense) passing through three sites of $\mathcal{S}$, and with empty interiors, that is to say the circles circumscribing an ordinary Delaunay triangle of $\mathcal{S}$ entirely drawn inside $I\!P$. An example of such a diagram on five points is given in Figure 9. Another example of hyperbolic Voronoï diagram in the model of the Poincaré circle is drawn in Figure 10.

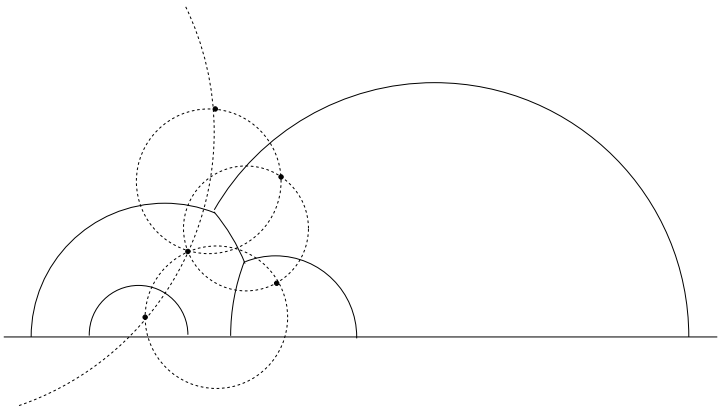

Figure 9: A Voronoï diagram in the hyperbolic Poincaré half plane

Now the discussion in Section 4 shows that the cells of the Delaunay triangulation of a set $\mathcal{S}$ of sites in two planes P and Q intersecting along l can be determined by superimposing two by two the four corresponding hyperbolic Voronoï diagrams : locating the vertices of one into the cells of the other will give the tetrahedra with three vertices in one plane, while those with two vertices in each plane will be given by intersections of edges. This is the exact analogue of paper[Boi88], the hyperbolic metric being replaced by the Euclidean one when P and Q are parallel.

Finally, it seems worth mentionning here that another, different, connexion between Delaunay triangulation (in the plane), and hyperbolic geometry (in space) has been already noticed and exploited by Igor Rivin in his thesis.[Riv86]

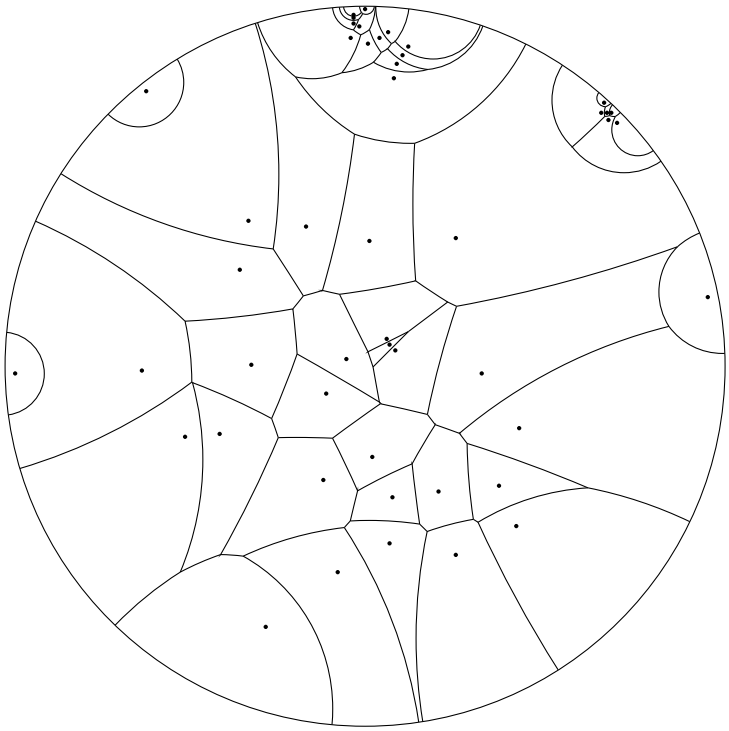

Figure 10: A Voronoï diagram in the hyperbolic Poincaré circle

Acknowledgements

O. Schwarzkopf is gratefully acknowledged for having provided reference to Matoušek[Mat91] and for helpful discussions which yielded to the higher dimensional result. The authors would also like to thank the referees for their comments that helped improving the presentation of the paper and Jean-Pierre Merlet for supplying him with his marvellous interactive drawing preparation system JPdraw .

References

[Aur91] F. Aurenhammer. Voronoi diagrams — a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3), September 1991.

[BDS*] J-D. Boissonnat, O. Devillers, R. Schott, M. Teillaud, and M. Yvinec. Applications of random sampling to on-line algorithms in computational geometry. *Discrete and Computational Geometry*. To appear. Available as Technical Report INRIA 1285. Abstract published in IMACS 91 in Dublin.

[Bea83] A. F. Beardon. *The geometry of discrete groups. Graduate texts in Mathematics*, Springer-Verlag, 1983.

[BM71] H. Bruggesser and P. Mani. Shellable decompositions of cells and spheres. *Math. Scand.*, 29:197–205, 1971.

[Boi88] J-D. Boissonnat. Shape reconstruction from planar cross-sections. *Computer Vision, Graphics, and Image Processing*, 44:1–29, 1988.

[Cox61] H.S.M. Coxeter. *Introduction to Geometry*. John Wiley & Sons, 1961.

[CS89] K.L. Clarkson and P.W. Shor. Applications of random sampling in computational geometry, II. *Discrete and Computational Geometry*, 4(5), 1989.

[Ede87] H. Edelsbrunner. *Algorithms on Combinatorial Geometry*. Springer-Verlag, 1987.

[EG86] H. Edelsbrunner and L.J. Guibas. Topologically sweeping an arrangement. In *ACM Symposium on Theory of Computing*, pages 389–403, 1986.

[EM85] H. Edelsbrunner and H. Maurer. Finding extreme points in three dimensions and solving the post office problem in the plane. *Information Processing Letters*, 21:39–47, July 1985.

[EM90] H. Edelsbrunner and E.P. Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9:66–104, January 1990.

[ES86] H. Edelsbrunner and R. Seidel. Voronoi diagrams and arrangements. *Discrete and Computational Geometry*, 1:25–44, 1986.

[GS86] L.J. Guibas and R. Seidel. Computing convolutions by reciprocal search. In *Second ACM Symposium on Computational Geometry in Yorktown Heights*, pages 90–99, June 1986.

[Mat91] J. Matoušek. Linear optimization queries. 1991. Manuscript.

[Ped70] D. Pedoe. *Geometry, a comprehensive course*. Dover Publications, New York, 1970.

[PS85] F.P. Preparata and M.I. Shamos. *Computational Geometry : an Introduction*. Springer-Verlag, 1985.

[Riv86] I. Rivin. *On geometry of convex polyhedra in hyperbolic 3-space*. PhD thesis, Computer Science Department, Princeton University, (USA), June 1986.

[Sei86] R. Seidel. Constructing higher-dimensional convex hulls at logarithmic cost per face. In *ACM Symposium on Theory of Computing*, pages 404–413, 1986.