Learning Smooth Shapes by Probing

Jean-Daniel Boissonnat
INRIA, 2004 route des lucioles
06902 Sophia-Antipolis
boissonn@sophia.inria.fr

Leonidas J. Guibas
Dept. Computer Science, Stanford U.
University. Stanford, CA 94305
guibas@cs.stanford.edu

Steve Oudot
INRIA, 2004 route des lucioles
06902 Sophia-Antipolis
soudot@sophia.inria.fr

Abstract

We consider the problem of discovering a smooth unknown surface S bounding an object $\mathcal{O}$ in $\mathbb{R}^3$. The discovery process consists of moving a point probing device in the free space around $\mathcal{O}$ so that it repeatedly comes in contact with S. We propose a probing strategy for generating a sequence of surface samples on S from which a triangulated surface can be generated that approximates S within any desired accuracy. We bound the number of probes and the number of elementary moves of the probing device. Our solution is an extension of previous work on Delaunay refinement techniques for surface meshing. The approximating surface we generate enjoys the many nice properties of the meshes obtained by those techniques, e.g. exact topological type, normal approximation, etc.

Keywords: Manifold learning, blind surface approximation, interactive surface reconstruction, surface meshing, Delaunay refinement

1 Introduction

A great deal of work in computational geometry and related communities has focussed on the problem of reconstructing a surface from scattered data points. The computational geometry community was the first to describe sampling conditions under which the geometry of the underlying surface can provably be approximated well and its topology fully recovered [2]. Testing if these sampling conditions are met, however, may require prior information about the surface that is not readily available or may be verified only after the fact (that is, after all the samples have been taken), if at all. As a result undesirable oversampling or undersampling may occur – in the former case sampling effort is wasted; in the latter provable reconstruction is impossible. In practice, the difficulty of testing the sampling conditions induces that the reconstruction algorithm is applied blindly, without any real mean to check the validity of the result.

A different and much less explored approach is to use the sampling conditions to guide the sampling process as the samples are being generated. Certain physical acquisition processes allow this type of fine control over the sampling process. One can think for instance of an autonomous robot moving in an unknown environment and coming repeatedly in contact with obstacles, where the aim is to learn enough about the environment so as to then be able to construct safe paths for the robot – see [15, 16] and the references therein. In this paper we consider the problem of discovering the shape of an unknown object $\mathcal{O}$ of $\mathbb{R}^3$ through an adaptive process of probing its surface from the exterior. A probe is issued along a ray whose origin lies outside $\mathcal{O}$ and returns the first point of $\mathcal{O}$ hit by the ray. Successive probes may require the probing device to be

moved through the free space outside $\mathcal{O}$. The goal is to find a strategy for the sequence of probes that guarantees a precise approximation of $\mathcal{O}$ after a minimal number of probes. Note that this problem involves an interesting bootstrapping issue, as the underlying surface is only known to the probing algorithm through the samples already taken. Thus, differently from most existing work in surface reconstruction, the data are not given all at once prior to the reconstruction phase but must instead be computed iteratively, each new probe depending on the outcomes of the previous probes. Furthermore, collision avoidance between the probing device and $\mathcal{O}$ must be observed at all times.

Given a surface of known positive reach (with a positive lower bound on its local feature size), the probing strategy proposed in this paper is inspired from Chew's algorithm [14] for Delaunay-based mesh refinement. Delaunay balls bounding surface facets are refined if they are too big. This refinement process is accomplished by moving our point probing device among current or prior edges of the dual Voronoi diagram known to lie in free space, before issuing a probe along the Voronoi edge dual to the facet to be refined. Our main contribution in this paper is the new probing algorithm proposed, the data structures used to find collision-free paths for the probing device, and the analysis of the total cost of this sampling procedure, including the number of probes made, the displacement cost for moving the device, and the combinatorial complexity of the construction. Our approach suggests numerous open problems that deserve further investigation.

1.1 Previous work

The above problem belongs to the class of geometric probing problems, pioneered by Cole and Yap [18]. Geometric probing, also known as blind approximation or interactive reconstruction, is motivated by applications in robotics. In this context, our probe model described above is called a tactile or finger probe. Geometric probing finds applications in other areas and gave rise to several variants. In particular, other probe models have been studied in the literature, e.g. line probes (a line moving perpendicular to a direction), X-ray probes (measuring the length of intersection between a line and the object), as well as their counterparts in higher dimensions.

We classify the probing algorithms into two main categories, exact or approximate, depending on whether they return the exact shape of the probed object or an approximation. An exact probing algorithm can only be applied to shapes that can be described by a finite number of parameters like polygons and polyhedra. In fact, most of the work on exact geometric probing is for convex polygons and polyhedra. See [34] for a survey of the computational literature on the subject. Although it has been shown that, using enhanced finger probes, a large class of non convex polyhedra can be exactly determined [1, 8], exact probing is too restrictive for most practical applications.

Approximate probing algorithms overcome this deficiency by considering the accuracy of the desired reconstruction as a parameter. The goal is to find a strategy that can discover a guaranteed approximation of the object using a minimal number of probes. The general problem is ill-posed, since we cannot conclude anything about the shape of the object if we have only local information about the shape. Some global information or prior knowledge is required to restrict the class of shapes being approximated. An important class is the class of convex shapes. Probing strategies have been proposed for planar convex objects using line probes [27, 31] and some other probe models are analyzed by Rote [32]. Observe that approximating a convex object using hyperplane probes is nothing else than approximating its supporting function.

As far as we know, probing non convex (non polyhedral) objects has not been studied. The problem has some similarity with surface approximation, where the goal is to construct a good

piecewise-linear (PL) approximation of a known smooth surface. Several provably good methods have been proposed to solve this problem. Some of them handle only restricted types of shapes, such as piecewise parametric CAD models [33, 37], solvent-excluded molecular surfaces [26], or skin surfaces [10, 11, 25]. Others hold in a more general setting but involve non-trivial geometric operations:

- The implicit surface mesher of Plantinga and Vegter [30] generates an adaptive grid and then applies a variant of the Marching Cubes algorithm [28]. Using interval arithmetics, Plantinga and Vegter can certify the topology of the output mesh $\hat{S}$. Moreover, by refining the grid sufficiently, they can achieve any given bound on the Hausdorff distance between $\hat{S}$ and S. This is a significant step since the Marching Cubes algorithm and its variants [13] usually come without any topological or geometric guarantees. However, the use of interval arithmetics requires to be able to compute the gradient of the function f whose zero-set is S.

- Algorithms based on the Closed Ball Property of Edelsbrunner and Shah [21], like the implicit surface mesher of Cheng *et al.* [12], require to be able to compute the critical points of height functions on the restrictions of S to some hyperplanes. The topology of the output mesh is ensured thanks to the Closed Ball Property.

- Methods based on critical points theory [5, 23] require to compute the critical points of f, and in some cases their indices, which is an even more evolved computation.

These geometric operations can be elegantly implemented in the implicit setting, where the surface is defined as a level set of some real-valued function, but not in the general case.

Differently, Chew's surface mesher [14] requires very little prior knowledge of S. Specifically, as emphasized in [7], it only needs to know S through:

1. a positive constant less than the reach of S,

2. an oracle that can tell whether a given line segment intersects S or not, and in the affirmative, return a point of intersection.

This oracle is strongly related to our probing model, yet surface probing differs from surface approximation in an essential way: we cannot place the probing device at will anywhere but need to plan the motion of the probing device to its next probing location. Differently from the convex case, we cannot simply probe from infinity and need to determine finite positions outside the object where to place the probing device. Moreover, in order to reach such positions, we need to determine paths along which the probing device can be safely moved without colliding with the object.

1.2 Statement of the problem

Let $\mathcal{O}$ be a bounded open set of $\mathbb{R}^3$ and S its boundary. The goal is to approximate S by a probing tool that can locate points on S. The following assumption allows us to localize $\mathcal{O}$ within $\mathbb{R}^3$, preventing indefinite searches.

A1 *For every connected component $\mathcal{O}_i$ of $\mathcal{O}$, we know a point o_i that belongs to $\mathcal{O}_i$.*

Assumption A2 bounds the area of interest and allows us to obtain initial locations and paths for the probing device without bumping into $\mathcal{O}$.

A2 *We know a convex and compact subset Ω of $\mathbb{R}^3$ that contains S (and hence also $\mathcal{O}$). We denote by $\partial\Omega$ the boundary of Ω.*

We have at our disposal a *probing device*, which is an oracle that, once placed at some point p of $\mathbb{R}^3 \setminus \mathcal{O}$, can be oriented towards any direction d and then tasked to return the first point of transverse intersection between S and the ray defined by (p, d). The probing device can move freely in $\mathbb{R}^3 \setminus \mathcal{O}$ but cannot penetrate $\mathcal{O}$. Such a device can be constructed in practice, using for instance a laser with three DOFs of displacement and two DOFs of rotation, that can cast a ray in any direction and measure its distance to the point where the ray hits the object.

We assume that the probing device provides *exact information*. The outcome of a probe is a point on the boundary of the object.

We need also to define the *accuracy measure* for our reconstruction. The accuracy will be measured by the Hausdorff distance. Since the measured points are on the boundary S of the object, the accuracy of the reconstruction will be ε iff any point of S is at distance at most ε from a measured point. In such a case, the set E of measured points is said to be an ε-sample of S.

As mentioned above, to be able to make any reconstruction claims, we need to restrict the class of shapes we probe. We consider here those with *positive reach*. The reach of a surface S, denoted by $\mathrm{rch}(S)$, is the infimum over S of the distance of a point of S to the medial axis of S. The reach has been previously used in many contexts and has received various names: reach [22], normal injectivity radius [9], minimum local feature size [2], etc. Having a positive reach is ensured if S is $C^{1,1}$, i.e. S is C^1 and its normal vector field is Lipschitz [22].

A3 *We know a positive constant $\varepsilon_S \leq \mathrm{rch}(S)$.*

Finally, we need a *model of computation* to analyze the complexity of our algorithm. Following the perception-action-cognition paradigm, we distinguish between the information or probing cost, the displacement cost, and the combinatorial cost. This distinction is also reminiscent of the difference made between combinatorial and informational complexity in the work on information-based computation [35, 36]. The probing cost measures the number of probes and indicates the amount of information that becomes available to our algorithm. The displacement cost accounts for the motion of the probing device. The combinatorial cost measures the arithmetic operations and comparisons required, as well as the maintenance cost of the data structures. As discussed later, it is not possible in general to optimize all costs simultaneously.

1.3 Overview of the paper

Under assumptions A1-A3, we show in this paper that S can be approximated by a triangulated surface $\hat{S}$ within any desired accuracy. Moreover, $\hat{S}$ recovers the exact topology of S and the error on the normal deviation of the facets of $\hat{S}$ is also bounded.

The paper is organized as follows. Since our solution is an extension of previous work on Delaunay refinement for surface meshing [7, 14], we recall Chew's algorithm and its main properties in Section 2. In Section 3 we describe the probing algorithm, present its main properties in Section 4, and analyze its complexity in Section 5. In these sections, the surface S is assumed to be connected, for simplicity. The case of a surface with more than one connected component is analyzed in Section 6.

2 Chew's algorithm

Chew's surface mesh generator is a greedy incremental algorithm that inserts sample points on S and maintains the *Delaunay triangulation* of the sample E *restricted* to S, defined below.

Input Chew's algorithm takes as input the surface S, a positive value ε, as well as an optional initial point sample. The surface is only known through an oracle ω that, given a line segment s, can compute a (possibly empty) subset of the intersection points of s with S.

Data structure Given a point set $E \subset S$, the Delaunay triangulation of E restricted to S, $\mathrm{Del}_{|S}(E)$, is the subset of the 3-dimensional Delaunay triangulation $\mathrm{Del}(E)$ of E made of the facets whose dual Voronoi edges intersect S. Every point of intersection of a Voronoi edge with S is the center of a *ball of* $\mathrm{Del}_{|S}(E)$, *i.e.* a Delaunay ball centered on S. By querying the oracle ω on every Voronoi edge, the algorithm can compute a subset of $\mathrm{Del}_{|S}(E)$, called $\mathrm{Del}^{\omega}_{|S}(E)$. Notice that $\mathrm{Del}^{\omega}_{|S}(E)$ may be different from $\mathrm{Del}_{|S}(E)$, since ω is not assumed to be able to detect all the intersection points of S with the edges of the Voronoi diagram. $\mathrm{Del}^{\omega}_{|S}(E)$ is stored as a subcomplex of $\mathrm{Del}(E)$. Each time a point is added to E, only the part of the Voronoi diagram that has changed after the insertion of the point has to be queried by the oracle ω.

Algorithm If no initial point sample E is given, the algorithm constructs one in the same way as our probing algorithm – see Section 3.1. $\mathrm{Del}^{\omega}_{|S}(E)$ is then computed by querying every edge of the Voronoi diagram of E using oracle ω.

At each iteration, the algorithm inserts a new point in E and updates $\mathrm{Del}^{\omega}_{|S}(E)$. Each point inserted in E is the center of a *bad ball of* $\mathrm{Del}^{\omega}_{|S}(E)$, that is, a ball of $\mathrm{Del}_{|S}(E)$ whose center c has been detected by ω and whose radius is greater than ε. The algorithm stops when there are no more bad balls of $\mathrm{Del}^{\omega}_{|S}(E)$, which will eventually happen if ε is positive since S is compact. Upon termination, the algorithm returns E as well as $\hat{S} = \mathrm{Del}^{\omega}_{|S}(E)$.

Guarantees on the output In [7], we proved that Chew's algorithm returns a triangulated complex $\hat{S}$ that is a good approximation of S, provided that the input parameter ε is smaller than a fraction of $\mathrm{rch}(S)$ and that the oracle can compute the intersection of any segment with S (such an oracle is said to be *exhaustive*). In fact, we proved a more general result, stated below as Theorem 2.1. This result holds under the following assumptions:

H1 *$\hat{S}$ is a manifold without boundary,*

H2 *$\hat{S}$ has vertices on all the connected components of S,*

H3 *Every facet f of $\hat{S}$ is circumscribed by a ball of $\mathrm{Del}_{|S}(E)$, of center $c \in S$ and of radius at most ε for $\varepsilon < 0.091\ \mathrm{rch}(S)$,*

Theorem 2.1 *Under H1-H3,*

- *$\hat{S}$ is ambient isotopic to S;*
- *the Hausdorff distance between $\hat{S}$ and S is at most $4.5\frac{\mathrm{diam}(S)}{\mathrm{rch}(S)^2}\,\varepsilon^2$;*
- *$\hat{S}$ approximates S, in terms of normals and area, within an error of $O(\varepsilon)$;*

– *S is covered by the balls of* $\mathrm{Del}_{|S}(E)$ *that circumscribe facets of* $\hat{S}$*, which implies that E is a* 2ε-sample *of S:* $\forall x \in S$, $|E \cap B(x, 2\varepsilon)| \geq 1$.

Moreover, it is proved in [7] that E is *sparse*: an r-sample E' is *sparse* if there is a constant κ that does not depend on S nor on r, such that $\forall x \in S$, $|E' \cap B(x, r)| \leq \kappa$. Thus,

Theorem 2.2 *[Theorems* 5.1 *and* 5.4 *of [7]]*
$|E| = \Theta\left(\iint_S \frac{dx}{\varepsilon^2}\right) = \Theta\left(\frac{\mathrm{Area}(S)}{\varepsilon^2}\right)$*, where the constants in the* Θ *do not depend on S nor on* ε.

In order to construct a PL approximation of S within a Hausdorff error of $\delta > 0$, it is sufficient to take:

$$\varepsilon = \varepsilon_S \sqrt{\frac{\delta}{4.5\ \mathrm{diam}(\Omega)}} \leq \mathrm{rch}(S) \sqrt{\frac{\delta}{4.5\ \mathrm{diam}(S)}}$$

The size of the output point set is then $O\left(\varepsilon^{-2}\right) = O\left(\delta^{-1}\right)$, which is optimal up to a constant depending only on S [17]. Note that the constraint on ε given by H3 yields a constraint on δ: $\delta < 0.04\ \mathrm{diam}(S)$.

3 The probing algorithm

For the sake of clarity, we assume in Sections 3, 4 and 5 that S is connected. We defer the treatment of surfaces with several connected components to Section 6. According to A1, we know a point $o \in \mathcal{O}$.

If we except the moves of the probing device, our algorithm is very similar to Chew's algorithm. The main difference concerns the oracle that is used to discover the surface S. In our case, to check whether a Voronoi edge e intersects S or not, we must first move our probing device to one of its endpoints. This requires two things: first, that at least one endpoint v of e be located in $\mathbb{R}^3 \setminus \mathcal{O}$; second, that we know a *free path* from $\mathbb{R}^3 \setminus \Omega$ (where the probing device can move freely) to v, *i.e.* a continuous curve included in $\mathbb{R}^3 \setminus \mathcal{O}$ that goes from $\mathbb{R}^3 \setminus \Omega$ to v.

Definition 3.1 *Given a point set E, the* Voronoi graph of E, $\mathrm{VG}(E)$*, is the graph made of the vertices and edges of the Voronoi diagram of E.*

Our basic intuition is to constrain the probing device to move along the edges of $\mathrm{VG}(E) \setminus \mathcal{O}$, which are called the *free edges*[1]. A difficulty arises from the fact that, when a new point p is inserted in E, some of the current Voronoi vertices and edges may disappear. It follows that portions of $\mathrm{VG}(E) \setminus \mathcal{O}$ that could be reached by the probing device from $\mathbb{R}^3 \setminus \Omega$ before the insertion of p may no longer be reachable afterwards — see Figure 1 for an illustration.

To overcome this difficulty, once a free path $\pi(v)$ from $\mathbb{R}^3 \setminus \Omega$ to some Voronoi vertex v has been found, we store $\pi(v)$ in memory so that v will remain reachable by the probing device permanently. Hence our paths are made of two types of edges: edges that belong to the current Voronoi graph, and edges that do not but were edges in some former Voronoi diagram.

By moving the probing device along our free paths, and by probing from each visited Voronoi vertex towards its neighbor vertices in $\mathrm{Vor}(E)$, we can detect a subset $\mathcal{I}$ of the points of $\mathrm{VG}(E) \cap S$ and construct a subcomplex of $\mathrm{Del}_{|S}(E)$ called the *visible restricted Delaunay triangulation of E*, or simply $\mathrm{Del}^v_{|S}(E)$. Every point of $\mathcal{I}$ is the center of a Delaunay ball, called *ball of* $\mathrm{Del}^v_{|S}(E)$, that circumscribes a facet of $\mathrm{Del}^v_{|S}(E)$.

[1] More generally, any object (point, segment, curve etc.) that lies outside $\mathcal{O}$ is said to be *free*.

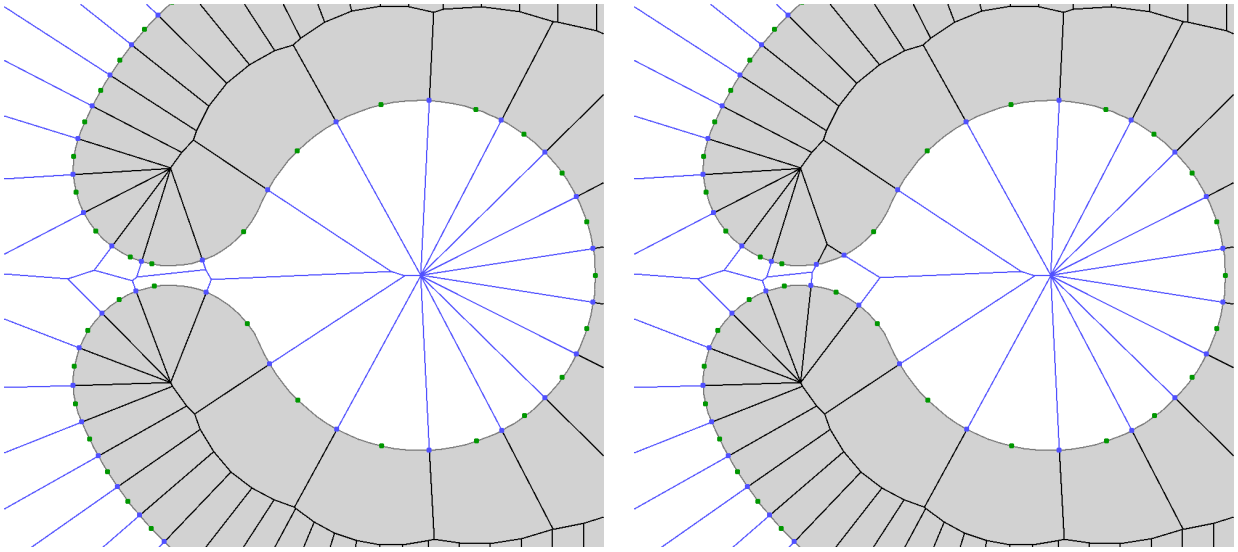

Figure 1: The insertion of a point splits $\mathrm{VG}(E) \setminus \mathcal{O}$ into two connected components, one of which is then no longer reachable from $\mathbb{R}^3 \setminus \Omega$.

3.1 Data structure

We proceed as in Chew's algorithm, by storing $\mathrm{Del}^v_{|S}(E)$ as a subcomplex of $\mathrm{Del}(E)$. Inside every Delaunay tetrahedron, we mark each of the four facets as being or not being part of $\mathrm{Del}^v_{|S}(E)$. This way, every Delaunay facet is marked twice since it belongs to two Delaunay tetrahedra.

In order to store the paths for the probing device, every Voronoi vertex[2] v is given a pointer *prev* to the previous vertex on a path from $\mathbb{R}^3 \setminus \Omega$ to v. By convention, $v.prev = NULL$ means that we know no free path from $\mathbb{R}^3 \setminus \Omega$ to v. In such a case, v is said to be *inactive*. Otherwise, v is called *active*.

If a newly created Voronoi vertex v belongs to $\mathbb{R}^3 \setminus \Omega$, then we set $v.prev \leftarrow v$ since v can be reached by the probing device. In particular, an infinite Voronoi vertex (*i.e.* the endpoint at infinity of an unbounded Voronoi edge) always lies outside Ω, which is compact. Thus, the *prev* field of an infinite vertex is never *NULL*. If v belongs to Ω, then we initialize $v.prev \leftarrow NULL$.

To construct and then update $\mathrm{Del}^v_{|S}(E)$, we use a routine named DETECT_ACCESS, introduced in Figure 2. Starting from an active vertex v_{start}, DETECT_ACCESS performs a depth-first traversal of $\mathrm{VG}(E) \setminus \mathcal{O}$ to see which previously inactive vertices can be reached by the probing device from v_{start} through free edges of the Voronoi graph.

Initial construction Given an initial point set E of S, we compute $\mathrm{Del}^v_{|S}(E)$ by moving the probing device successively to all the vertices of $\mathrm{VG}(E)$ that lie outside Ω (including the infinite vertices[3]). For every such vertex v, we set $v.prev \leftarrow v$ and then we call DETECT_ACCESS on v.

[2] In practice, it is its dual Delaunay tetrahedron that we consider. However, for simplicity, we will identify Delaunay tetrahedra with Voronoi vertices in the sequel.

[3] Processing the infinite vertices in the same manner as the other ones simplifies the presentation but is not quite satisfactory since it involves moving the probing device to infinity. However, this can be avoided easily by clipping $\mathrm{VG}(E)$ by $\partial\Omega$ and calling DETECT_ACCESS on all the intersection points of $\partial\Omega \cap \mathrm{VG}(E)$.

```
DETECT_ACCESS (v_start):
 // Precondition: v_start is active
for each neighbor v of v_start, do
    PROBE edge [v_start, v];
    if ([v_start, v] ∩ S ≠ ∅) then
        add the dual of [v_start, v] to Del^v_|S(E);
    else if (v.prev = NULL) then
      // v becomes active
        if (v ∈ Ω) then
            v.prev ← v_start;
        end if
        MOVE the probing device from v_start to v;
        DETECT_ACCESS (v);
        MOVE the probing device from v to v_start;
    end if
end foreach
```

Figure 2: Routine DETECT_ACCESS

After the initialization phase, every Voronoi vertex that can be reached from $\mathbb{R}^3 \setminus \Omega$ by walking along edges of $\mathrm{VG}(E) \setminus \mathcal{O}$ is active. Moreover, every active vertex is given a free path to $\mathbb{R}^3 \setminus \Omega$.

Update Each time a new point p is to be inserted in E, we update $\mathrm{Del}^v_{|S}(E)$ as follows:
• before the insertion, we look at the active vertices of $\mathrm{Vor}(E)$ that no longer exist in $\mathrm{Vor}(E \cup \{p\})$. By definition, they lie in $\mathrm{V}(p)$, the cell of p in $\mathrm{Vor}(E \cup \{p\})$. We keep these vertices in memory and we leave their *prev* pointers unchanged. This way, every active vertex will remain active in the sequel and will keep its path to $\mathbb{R}^3 \setminus \Omega$.
• after the insertion, we look at the new vertices of the Voronoi diagram (including the infinite ones), which by definition are the vertices of $\mathrm{V}(p)$. For any such vertex v, we need to determine whether v can be reached from $\mathbb{R}^3 \setminus \Omega$ through edges of $\mathrm{VG}(E) \setminus \mathcal{O}$:
- if $v \in \mathbb{R}^3 \setminus \Omega$, we set $v.prev \leftarrow v$ and move the probing device to v. Such a move is called a *positioning displacement*. Then, we call DETECT_ACCESS on v.
- otherwise, we look at the only neighbor v' of v that is not a vertex of $\mathrm{V}(p)$. If v' is active and if edge $[v, v']$ is free (which we can easily determine since $[v, v']$ is included in a former Voronoi edge that has been probed from v'), we perform a *positioning displacement* by moving the probing device to v'. Then, we call DETECT_ACCESS on v'.

3.2 The algorithm

The algorithm takes as input a user-defined value ε such that $0 < \varepsilon < 0.091\ \varepsilon_S$, which by A3 is less than $0.091\ \mathrm{rch}(S)$. As explained in Section 2, controlling ε allows to bound the Hausdorff distance between S and the PL approximation built by the algorithm.

The algorithm starts by computing an initial point set E made of three points of S that form a triangle of circumradius at most $\varepsilon/3$. There are many ways to do this. One possible approach

is described in detail in [29, §4.4] in the context of surface meshing, and it extends easily to the context of surface probing. Here is a high-level overview:

1. We place the probing device at a point p of $\partial\Omega$ and we probe from p towards point o. Since $o \in \mathcal{O}$ and $p \in \mathbb{R}^3 \setminus \mathcal{O}$, the probing device finds a point $a \in S$, such that the segment $[p, a]$ is free.

2. Suppose we knew the normal $\mathbf{n}(a)$ of S at a, or at least a good estimate $\mathbf{n}$. Then, we could move the probing device from p to a, and then along the free section of the ray issued at a in the direction of $\mathbf{n}$. Let a' be some point on the free section of the ray, and T' be the plane containing a' and parallel to the tangent plane of S at a. Inside T', we can move the probing device from a' to two arbitrarily close points b', c' that form an equilateral triangle with a'. Probing from b' and c' towards $-\mathbf{n}$ gives two points $b, c \in S$, such that triangle (a, b, c) is almost equilateral, as shown in [29, §4.4]. If b', c' are chosen sufficiently close to a', then (a, b, c) has a circumradius less than $\varepsilon/3$.

3. In the context of surface probing, if the probing device provides points and normals, then we can apply step 2. directly. Otherwise, we do not know the normal of S at a but we can approximate it. Standing at p, we probe in a direction arbitrarily close to $[p, a)$, which gives a point $a'' \in S$ such that the distance between a and a'' is at most ε – this condition can be easily checked since ε is a known parameter. Then, the bisector plane P of $[a, a'']$ contains a direction that approximates $\mathbf{n}(a)$ within an angle of $O(\varepsilon)$. We do not know this direction, but we can approximate it by moving the probing device from p to a and then performing a sequence of probes inside the plane P_a parallel to P that contains a. Specifically, we probe along a set of directions that forms an $O(\varepsilon)$-net of the unit circle centered at a inside P_a, and we select the directions whose rays intersect S at a (or at some point very close to a, in practice). Then, we define $\mathbf{n}$ as the direction (on the unit circle) farthest from the set of selected directions.

   Intuitively, since P_a is almost aligned with $\mathbf{n}(a)$, the curve $P_a \cap S$ has a small curvature at a, compared to $1/\varepsilon$. Therefore, the set of selected directions spans approximately half of the unit circle, and the direction $\mathbf{n}$ farthest from the selected directions approximates $\mathbf{n}_{P_a}$ within an angle of $O(\varepsilon)$. Since $\mathbf{n}_{P_a}$ is aligned with the orthogonal projection of $\mathbf{n}(a)$ onto P_a, the angle $(\mathbf{n}_{P_a}, \mathbf{n}(a))$ is $O(\varepsilon)$. It follows that $(\mathbf{n}, \mathbf{n}(a)) = O(\varepsilon)$. We can then apply step 2. above to compute b and c, using $\mathbf{n}$.

Once the initial point sample $\{a, b, c\}$ is computed, the algorithm sets $E = \{a, b, c\}$ and builds $\mathrm{Del}^v_{|S}(E)$ as described in Section 3.1. Since (a, b, c) is the only facet of $\mathrm{Del}(E)$, it belongs to $\mathrm{Del}^v_{|S}(E)$. Moreover, as shown in [7] (Lemma 7.1), (a, b, c) will remain in $\mathrm{Del}_{|S}(E)$ throughout the process[4]. For this reason, we call it a *persistent facet*. The *bad* balls of $\mathrm{Del}^v_{|S}(E)$, *i.e.* the balls of $\mathrm{Del}^v_{|S}(E)$ whose radii are greater than ε, are stored in a priority queue $\mathcal{Q}$ where they are sorted by decreasing radius.

After the initialization phase, the algorithm acts as Chew's surface mesher, using the probing device to answer the oracle. Specifically, the data structure is $\mathrm{Del}^v_{|S}(E)$, and the bad balls of $\mathrm{Del}^v_{|S}(E)$ are stored in $\mathcal{Q}$. While $\mathcal{Q}$ is not empty, the algorithm retrieves from $\mathcal{Q}$ the bad ball $B(c, r)$ of largest radius and inserts its center c in E. The algorithm then updates $\mathrm{Del}^v_{|S}(E)$ as described in Section 3.1, and updates $\mathcal{Q}$ as follows:

[4]Notice however that (a, b, c) is not guaranteed to remain in $\mathrm{Del}^v_{|S}(E)$.

– the former bad balls that disappear because of the insertion of c are removed from $\mathcal{Q}$;
– the new bad balls that are created by the insertion of c are inserted in $\mathcal{Q}$.

The algorithm stops when $\mathcal{Q}$ is empty, that is, when no ball of $\mathrm{Del}^v_{|S}(E)$ is bad. The algorithm then returns E and $\mathrm{Del}^v_{|S}(E)$.

4 Correctness of the algorithm and quality of the approximation

In this section, we analyze the probing algorithm. We prove that it terminates in Section 4.1. In Section 4.2, we exhibit two invariants that are instrumental in proving the geometric properties of the output surface in Section 4.3. The analysis of the complexity of the algorithm is deferred to Section 5.

4.1 Termination

After the initialization phase, every point that is inserted in E belongs to S and is the center of a Delaunay ball of radius greater than ε. It follows that the points inserted in E are at distance at least ε from one another. Since ε is positive and S is compact, only finitely many points are inserted in E.

4.2 Invariants of the algorithm

Proposition 4.1 *The following assertions hold throughout the course of the algorithm:*
P1 *All active Voronoi vertices can be reached from* $\mathbb{R}^3 \setminus \Omega$ *by moving the probing device along current or former Voronoi edges.*
P2 *Any two Voronoi vertices that lie in the same connected component of* $\mathrm{VG}(E) \setminus \mathcal{O}$ *have the same status,* active *or* inactive.

Proof We proceed by induction. Clearly, (P1) and (P2) hold after the initialization phase. Let us now consider a step of the algorithm during which a new point (say p) is inserted in E and $\mathrm{Del}^v_{|S}(E)$ is updated. Our induction hypothesis is the following:

IH *Assertions (P1) and (P2) hold in set E before the insertion of p.*

We will prove successively that (P1) and (P2) still hold after the insertion of p. In the sequel, E denotes the point sample before the insertion of p.

(P1) Let v be a vertex that is active after the insertion of p.
P1.1 If v existed and was already active before the insertion of p, then its path $\pi(v)$ to $\mathbb{R}^3 \setminus \Omega$ remains unchanged since all the vertices of $\pi(v)$ are kept in memory and DETECT_ACCESS does not change the status of active vertices. It follows that v is reachable by the probing device from $\mathbb{R}^3 \setminus \Omega$ after the insertion of p, since it was so before by (IH).
P1.2 If v did not exist or was not active before the insertion of p, then v is visited by DETECT_ACCESS during the update of $\mathrm{Del}^v_{|S}(E)$. Since we run DETECT_ACCESS only on new vertices lying in $\mathbb{R}^3 \setminus \Omega$ and on former active vertices, v is given a free path either to a new vertex lying in $\mathbb{R}^3 \setminus \Omega$, or to a former active vertex which, as explained in P1.1, remains reachable by the probing device after the insertion of p. In both cases, v is reachable by the probing device from $\mathbb{R}^3 \setminus \Omega$.

(P2) Let us prove that the vertices v and w of any free edge e of $\mathrm{VG}(E \cup \{p\})$ have the same status after the update of $\mathrm{Del}^v_{|S}(E)$. It will then follow, by transitivity, that (P2) still holds after the insertion of p.
P2.1 If a vertex of e (say v) is visited by DETECT_ACCESS during the update of $\mathrm{Del}^v_{|S}(E)$, then it becomes active if not so before, and DETECT_ACCESS visits also w if the latter is not active. Thus, v and w are both active afterwards.
P2.2 If neither v nor w is visited by DETECT_ACCESS, then they keep their status during the update of $\mathrm{Del}^v_{|S}(E)$. Hence, it suffices to prove that they have the same status right before. If neither v nor w is a vertex of $\mathrm{V}(p)$, then they are both old Voronoi vertices, and e is an old edge, which implies that v and w have the same status, by (IH). If one of them belongs to $\mathrm{V}(p)$, then none can be active, since otherwise, during the update of $\mathrm{Del}^v_{|S}(E)$, the algorithm would run DETECT_ACCESS on the one(s) that is (are) active, hereby contradicting the hypothesis of P2.2. □

4.3 Geometric properties of the output

As explained in Section 2, in order to guarantee that the algorithm constructs a good approximation of S, it suffices to prove that $\mathrm{Del}^v_{|S}(E)$ satisfies assertions (H1), (H2) and (H3) upon termination of the algorithm. From now on, let E denote the output point sample.

Proof of H2 Since we assumed that S is connected, it suffices to check that $\mathrm{Del}^v_{|S}(E)$ is not empty when the algorithm halts. Recall that the algorithm constructs an initial point sample with a *persistent facet* (a, b, c) cicumscribed by a Delaunay ball B centered on S of radius at most $\varepsilon/3$. As shown in [7] (Lemma 7.1), (a, b, c) remains a facet of $\mathrm{Del}_{|S}(E)$ throughout the course of the algorithm. It follows that $\mathrm{VG}(E) \cap S$ is not empty upon termination of the algorithm. Since $\mathrm{VG}(E)$ is connected, at least one point p of $\mathrm{VG}(E) \cap S$ belongs to the same connected component of $\mathrm{VG}(E) \setminus \mathcal{O}$ as some infinite Voronoi vertex. By (P2), p can be "seen" from an active Voronoi vertex. Hence, $\mathrm{Del}^v_{|S}(E)$ is not empty, which proves (H2). □

Proof of H3 By definition, every facet of $\mathrm{Del}^v_{|S}(E)$ is circumscribed by a ball of $\mathrm{Del}^v_{|S}(E)$. Since the algorithm eliminates the balls of $\mathrm{Del}^v_{|S}(E)$ that have radii greater than ε, all the balls of $\mathrm{Del}^v_{|S}(E)$ have radii at most $\varepsilon < 0.091\ \varepsilon_S$ upon termination. By A3, ε is less than $0.091\ \mathrm{rch}(S)$. □

As established in Section 3 of [7], assertion (H3) alone induces a few local properties[5], such as:

L1 [Lemma 3.4 of [7]] *Two facets of* $\mathrm{Del}^v_{|S}(E)$ *that share an edge form a dihedral angle greater than* $\frac{\pi}{2}$.

L2 [Lemma 3.6 of [7]] *An edge of* $\mathrm{Vor}(E)$ *cannot intersect* S *in more than one point* x *such that* $\mathrm{dist}(x, E) < 0.091\ \mathrm{rch}(S)$. *Hence, every edge of* $\mathrm{Vor}(E)$ *contains at most one center of ball of* $\mathrm{Del}^v_{|S}(E)$.

L3 [Proposition 3.10 of [7]] *The balls of* $\mathrm{Del}^v_{|S}(E)$ *intersect* S *along* pseudo-disks, *i.e. topological disks that pairwise intersect along topological disks and whose boundaries pairwise intersect in at most two points.*

[5] Propositions L1 and L2 can also be inferred from the results of [3, 10].

To prove (H1), we need yet another result, which is a direct consequence of assertion (P2):

L4 *Let ζ be a connected component of $\mathrm{VG}(E) \setminus \mathcal{O}$. Either all the points of $\partial\zeta \cap S$ are centers of balls of $\mathrm{Del}^v_{|S}(E)$, or none of them is.*

Proof Let p and q be two points of $\partial\zeta \cap S$. By definition, p and q are centers of balls of $\mathrm{Del}_{|S}(E)$. If ζ contains no (finite or infinite) Voronoi vertex, then it is made of one piece of a Voronoi edge only. Therefore, p and q cannot be detected by the probing device, and none of them can be the center of a ball of $\mathrm{Del}^v_{|S}(E)$. If ζ contains some Voronoi vertices, then, by (P2), all the Voronoi vertices in ζ have the same status, *active* or *inactive*. In the first case, p and q are both centers of balls of $\mathrm{Del}^v_{|S}(E)$. In the second case, none of them is, which ends the proof of (L4). □

Using L1-L4, we can now prove Assertion (H1).

Proof of H1 We first show that every edge of $\mathrm{Del}^v_{|S}(E)$ is incident to exactly two facets of $\mathrm{Del}^v_{|S}(E)$. We then prove that every vertex of $\mathrm{Del}^v_{|S}(E)$ has only one *umbrella*. An umbrella of a vertex v is a set of facets of $\mathrm{Del}^v_{|S}(E)$ incident to v whose adjacency graph is a cycle.

Let e be an edge of $\mathrm{Del}^v_{|S}(E)$. We denote by $\mathrm{V}(e)$ the Voronoi facet dual to e. Notice that $\partial\mathrm{V}(e) \cap S \neq \emptyset$, since e belongs to $\mathrm{Del}_{|S}(E)$. It follows that any connected component ξ of $\partial\mathrm{V}(e) \setminus \mathcal{O}$ is a simple polygonal arc, whose endpoints lie on S and are centers of balls of $\mathrm{Del}_{|S}(E)$. Moreover, ξ is included in a connected component of $\mathrm{VG}(E) \setminus \mathcal{O}$. Thus, by (L4), either both endpoints of ξ are centers of balls of $\mathrm{Del}^v_{|S}(E)$, or none of them is. It follows that the total number of centers of balls of $\mathrm{Del}^v_{|S}(E)$ that lie on $\partial\mathrm{V}(e)$ is even. Then, by (L2), the number of edges of $\partial\mathrm{V}(e)$ that contain centers of balls of $\mathrm{Del}^v_{|S}(E)$ is even. Equivalently, the number of facets of $\mathrm{Del}^v_{|S}(E)$ that are incident to e is even.

In addition, two facets of $\mathrm{Del}^v_{|S}(E)$ incident to e form a dihedral angle greater than $\frac{\pi}{2}$, by (L1). It follows that e cannot be incident to more than three facets of $\mathrm{Del}^v_{|S}(E)$.

In conclusion, the number of facets of $\mathrm{Del}^v_{|S}(E)$ incident to e is even, at least 1 (because e is an edge of $\mathrm{Del}^v_{|S}(E)$), and at most 3. Hence it is 2.

Since this is true for any edge of $\mathrm{Del}^v_{|S}(E)$, the facets of $\mathrm{Del}^v_{|S}(E)$ incident to a given vertex v of $\mathrm{Del}^v_{|S}(E)$ form a set of umbrellas. Using (L3), one can prove that they form only one umbrella – see Proposition 4.2 of [7]. We recall briefly the argument: if U is an umbrella, then v lies in the interior of the projection of U onto $T(v)$, since otherwise the projections of at least two adjacent facets of U would have non-disjoint interiors, which would imply by (L3) that one of their vertices lies inside one of their pseudo-disks, which is impossible since the pseudo-disks are empty of points of E. It follows that v belongs to the interior of the union R of the pseudo-disks of the facets of U, by (L3). Then, any facet f of $\mathrm{Del}^v_{|S}(E)$ incident to v that does not belong to U has one vertex (namely, v) that belongs to the intorior of R, whereas its two other vertices lie outside R. Hence, the boundary of the pseudo-disk of f intersects the boundary of R. Using (L3) again, it is not difficult to prove that the pseudo-disk of f contains at least one vertex of U, which contradicts the fact that the pseudo-disks are empty of points of E.

Therefore, a vertex of $\mathrm{Del}^v_{|S}(E)$ can have only one umbrella. It follows that $\mathrm{Del}^v_{|S}(E)$ is a 2-manifold without boundary, which concludes the proof of H1. □

Since $\mathrm{Del}^v_{|S}(E)$ satisfies H1-H3, it is a good approximation of S, according to Theorem 2.1. In particular, E is a 2ε-sample of S, and it is sparse, as mentioned in Section 2. This implies

that $|E| = O\left(\frac{\text{Area}(S)}{\varepsilon^2}\right)$, by Theorem 2.2. Moreover, if $\varepsilon < 0.05\ \varepsilon_S$, then E is a 0.1-sample of S, and hence $\text{Del}_{|S}(E)$ is homeomorphic to S, by Theorem 2 of [2]. As a consequence, $\text{Del}^v_{|S}(E)$ and $\text{Del}_{|S}(E)$ are equal, since they are homeomorphic and since $\text{Del}^v_{|S}(E)$ is a subcomplex of $\text{Del}_{|S}(E)$.

5 Complexity of the algorithm

As mentioned in the introduction, the complexity of the algorithm has three components: the *combinatorial cost* that measures the memory space and time needed to store, construct and update the data structures; the *probing cost* that counts the number of probes performed by the probing device; the *displacement cost* that measures the effort spent in moving the probing device. Depending on the context, one can give emphasis to one type of cost or the other.

Notice that it is not possible in general to optimize all costs simultaneously. Take for instance a parabola $\mathcal{C}$ embedded in $\mathbb{R}^2$, as shown in Figure 3. Any Delaunay-based algorithm that optimizes the displacements of the probing device will somehow follow the curve $\mathcal{C}$, inserting the points of E more or less in their order along $\mathcal{C}$ (see Figure 3, left). This makes the overall complexity of the incremental Delaunay triangulation quadratic. Differently, our algorithm will insert the points in an order defined by the *largest empty ball* criterion (see Figure 3, right), which does not optimize the displacement cost but makes the combinatorial cost linear.

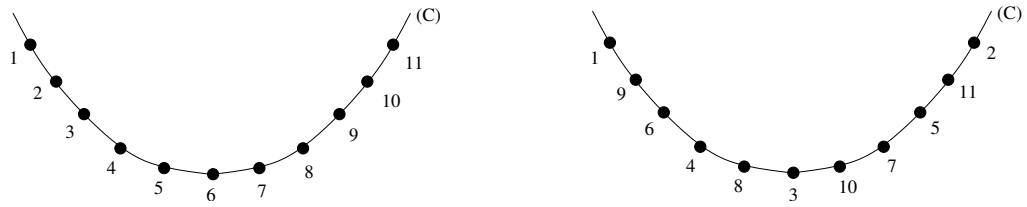

Figure 3: Two orders of insertion on a parabola.

In the sequel, we analyze the combinatorial cost, probing cost and displacement cost separately. Since our algorithm enforces the probing device to move along the Voronoi edges, the size of the Voronoi diagram has a direct impact on all three costs.

5.1 Combinatorial cost

Space complexity

The data structure stores the current Delaunay triangulation as well as some of the former Voronoi vertices. Since every vertex is stored at most once, the size of the data structure is at most the total number of Voronoi vertices created during the course of the algorithm. We will bound this number with respect to the Hausdorff distance δ between $\hat{S}$ and S.

Let E_{init} be the initial point sample constructed by the algorithm. We have $|E_{\text{init}}| = 3$. For every iteration i of the algorithm, we call $E(i)$ the point set E at the end of iteration i. $E(i)\backslash E(i-1)$ contains precisely the point $p(i)$ inserted in E at iteration i, and $E(i-1) \setminus E_{\text{init}}$ is the set of all points inserted before iteration i. We call $r(i)$ the radius of the largest ball of $\text{Del}^v_{|S}(E(i))$. Since the algorithm always inserts the center of the ball of $\text{Del}^v_{|S}(E)$ of largest radius, $p(i)$ is at a distance $r(i-1)$ from $E(i-1)$.

Let Z be the subset of the ridge of S made of all the points of S that admit an osculating ball whose interior does not intersect S. We assume in the sequel that Z is a set of curves of finite length. As mentioned in [4], this property of Z is satisfied generically. In particular, S cannot contain patches of spheres or cylinders with empty osculating spheres. To bound the number of Voronoi vertices created, we will use the following result, stated as Lemma 17 in [4]:

Lemma 5.1 *There exist four constants ε_0, c_0, k_1 and k_2, depending only on S, such that, for any sparse ε-sample E of S, with $\varepsilon \leq \varepsilon_0$, the number of Delaunay edges incident to a vertex p of* $\mathrm{Del}(E)$ *is at most $k_1\, \varepsilon^{-1/2}$ if $\mathrm{dist}(p, Z) \leq c_0\, \sqrt{\varepsilon}$ and at most k_2 / $\mathrm{dist}(p, Z)$ otherwise.*

In the sequel, we take as ε_0 the minimum of the above (unknown) constant ε_0 and of $0.091\ \mathrm{rch}(S)$. Let i_0 be the first iteration of the algorithm at the end of which all the balls of $\mathrm{Del}^v_{|S}(E)$ have radii at most $\frac{\varepsilon_0}{4}$. In other words, i_0 is the first iteration such that $r(i_0) \leq \frac{\varepsilon_0}{4}$. Since $\frac{\varepsilon_0}{4} < 0.091\ \mathrm{rch}(S)$, $E(i_0)$ is an $\frac{\varepsilon_0}{2}$-sample of S, by Theorem 2.1.

Lemma 5.2 *For any two iterations i and j such that $j \geq i \geq i_0$, we have $r(j) \leq 2r(i)$.*

Proof Let $B(j)$ be a ball of $\mathrm{Del}^v_{|S}(E(j))$ of largest radius. Its center $c(j)$ lies on S. Since $i \geq i_0$, we have $E(i_0) \subseteq E(i)$. Hence, $E(i)$ is an ε_0-sample of S, and the balls of $\mathrm{Del}^v_{|S}(E(i))$ cover S, by Theorem 2.1. Thus, $c(j)$ lies in a ball $B(c, r)$ of $\mathrm{Del}^v_{|S}(E(i))$. We have $\mathrm{dist}(c(j), E(i)) \leq 2r \leq 2r(i)$. Moreover, since $i \leq j$, $E(i)$ is included in $E(j)$. It follows that $r(j) = \mathrm{dist}(c(j), E(j)) \leq \mathrm{dist}(c(j), E(i)) \leq 2r(i)$, which concludes the proof of the lemma. □

Lemma 5.3 *For any iteration $i > i_0$, $E(i)$ is a $2r(i)$-sample of S, with $2r(i) \leq \varepsilon_0$, and the points of $E(i) \setminus E_{init}$ are farther than $\frac{r(i-1)}{2}$ from one another and from E_{init}.*

Proof Let i be any iteration of the algorithm such that $i > i_0$. According to Lemma 5.2, we have $r(i) \leq 2r(i_0) \leq \frac{\varepsilon_0}{2}$, thus $E(i)$ is a $2r(i)$-sample of S, with $2r(i) \leq \varepsilon_0$, according to Theorem 2.1. In addition, by definition of i_0, every point of $E(i_0) \setminus E_{\mathrm{init}}$, when inserted in E, is the center of a Delaunay ball of radius greater than $\frac{\varepsilon_0}{4} \geq r(i_0)$, which is at least $\frac{1}{2}\, r(i-1)$, by Lemma 5.2. Moreover, at any iteration k such that $i_0 < k \leq i$, the point inserted in E is the center of a Delaunay ball of radius $r(k-1)$, which is at least $\frac{1}{2}\, r(i-1)$, by Lemma 5.2. Therefore, the points of $E(i) \setminus E_{\mathrm{init}}$ are at least $\frac{1}{2}\, r(i-1)$ away from one another and from E_{init}. □

From the fact that $|E_{\mathrm{init}}| = 3$ and from Lemmas 5.2 and 5.3, we deduce that for any $i > i_0$, $E(i)$ is a sparse $2r(i)$-sample of S. Thus, by Lemma 5.1, the algorithm creates $O\left(r(i)^{-1/2}\right) = O\left(\varepsilon^{-1/2}\right)$ new Delaunay edges at iteration i. As a consequence, the overall number of Delaunay edges created after iteration i_0 (which depends only on S) is $O\left(N\, \varepsilon^{-1/2}\right) = O\left(N^{5/4}\right)$, where N is the size of the output point sample and where the constant in the O depends only on S.

However, by summing more carefully the contributions of the points inserted after iteration i_0, we can work out a $O\left(N \log N\right)$ bound. Let i be an iteration of the algorithm, such that $i \geq i_0$. Let $j > i$ be the first iteration such that $r(j) \leq \frac{r(i)}{8}$. Our goal is to bound the number of Delaunay edges created between iterations i and j. By Lemma 5.3, $E(j)$ is a $2r(j)$-sample of S, with $2r(j) \leq \frac{r(i)}{4}$. We call $E(i, j)$ the set of the points inserted by the algorithm between iterations i (excluded) and j (included). We have $E(i, j) = E(j) \setminus E(i)$.

Lemma 5.4 *For any k such that $i < k \leq j$, $E(k)$ is a sparse $6r(i)$-sample of S.*

Proof By Lemma 5.3, $E(k)$ is a $2r(k)$-sample of S. Since $2r(k) \leq 4r(i) \leq 6r(i)$ (Lemma 5.2), $E(k)$ is a $6r(i)$-sample. To prove that $E(k)$ is sparse, we count the points of $E(k)$ that lie in $B(x, 6r(i))$, for any $x \in S$.
- Since $|E_{\text{init}}| = 3$, the number of points of E_{init} that lie in $B(x, 6r(i))$ is at most 3.
- By Lemma 5.3, the points of $E(k) \setminus E_{\text{init}}$ are farther than $\frac{r(k-1)}{2}$ from one another. Now, $\frac{r(k-1)}{2}$ is at least $\frac{r(i)}{16}$, since $i < k \leq j$. It follows that the points of $E(k) \setminus E_{\text{init}}$ are centers of pairwise-disjoint balls of radius $\frac{1}{32}\, r(i)$. For every such ball B whose center lies in $B(x, 6r(i))$, B is included in $B(x, (6 + \frac{1}{32})\, r(i))$. It follows that the number of points of $E(k) \setminus E_{\text{init}}$ that lie in $B(x, 6r(i))$ is bounded by a constant, which shows that $E(k)$ is sparse and hereby concludes the proof of Lemma 5.4. □

Lemma 5.5 *$E(i, j)$ is a sparse $6r(i)$-sample of S.*

Proof Let u be a point of $E(i + 1)$. By Corollary 4.13 of [7], u is a vertex of $\text{Del}_{|S}(E(i + 1))$. $\text{Del}_{|S}(E(i + 1))$ is a 2-manifold without boundary, thus u has at least three neighbors v_1, v_2, v_3 in $\text{Del}_{|S}(E(i+1))$. Since $|E_{\text{init}}| = 3$, at least one point among $\{u, v_1, v_2, v_3\}$ belongs to $E(i+1) \setminus E_{\text{init}}$. Let us call it w. By Lemma 5.3, w is farther than $\frac{r(i)}{2}$ from the other points of $\{u, v_1, v_2, v_3\}$. Thus, u is farther than $\frac{r(i)}{2}$ from one of its neighbors, say v_1. Any point $x \in S$ belonging to the Voronoi face $\text{V}(u) \cap \text{V}(v_1)$ is farther than $\frac{r(i)}{4}$ from u. Hence, x is closer to some point u' of $E(i+1, j)$ than to u, since otherwise $E(j)$ could not be a $\frac{r(i)}{4}$-sample of S. As a consequence, the distance from any point $y \in S \cap \text{V}(u)$ to $E(i + 1, j)$ is at most:

$$\begin{aligned} \text{dist}(y, u') &\leq \text{dist}(y, u) + \text{dist}(u, x) + \text{dist}(x, u') \\ &\leq \text{dist}(y, u) + 2\, \text{dist}(u, x) \end{aligned}$$

Since $E(i + 1)$ contains $E(i)$, $E(i + 1)$ is a $2r(i)$-sample of S. Thus, $\text{dist}(y, u) \leq 2r(i)$ and $\text{dist}(u, x) \leq 2r(i)$, which implies that $\text{dist}(y, E(i + 1, j)) \leq 6r(i)$. Since this is true for any $u \in E(i + 1)$, $E(i + 1, j)$ is a $6r(i)$-sample of S. So is $E(i, j)$, for $E(i, j) \supset E(i + 1, j)$.

In addition, by definition of j, any point of $E(i, j)$, right before its insertion, is the center of a Delaunay ball of radius greater than $\frac{r(i)}{8}$. It follows that the points of $E(i, j)$ are farther than $\frac{r(i)}{8}$ from one another. Hence, by the same argument as in the proof of Lemma 5.4, $E(i, j)$ is sparse. □

We can now combine Lemma 5.1 with Lemmas 5.4 and 5.5, to bound the number of Delaunay edges created between iterations i and j.

Lemma 5.6 *During the insertion of the points of $E(i, j)$, the number of Delaunay edges created is $O\left(|E(i, j)| \log |E(i, j)|\right)$.*

Proof Let $\varepsilon_i = 6r(i)$. The reasoning is similar in spirit to that of Lemma 18 of [4], although with an additional subtelty. We decompose S into strips parallel to Z, of width $c_0\sqrt{\varepsilon_i}$, where c_0 is defined as in Lemma 5.1. Recall that c_0 depends on S but not on ε_i. Let Z_k denote the k^{th} strip ($k \geq 0$). The points of Z_k lie at a distance of Z ranging from $k\, c_0\sqrt{\varepsilon_i}$ to $(k + 1)\, c_0\sqrt{\varepsilon_i}$.

As stated in Lemma 18 of [4], since $E(i,j)$ is a sparse ε_i-sample of S (Lemma 5.5), there exists some constant $c(S)$ depending only on S, such that the number of points of $E(i,j)$ that lie in a given strip Z_k is at most $c(S)\ \varepsilon_i^{-3/2}$. Moreover, for any i' such that $i < i' \leq j$, $E(i')$ is a sparse ε_i-sample of S (Lemma 5.4), thus Lemma 5.1 applies to the point inserted at iteration i'. Summing the contributions of all the points of $E(i,j)$, we find that the number n of Delaunay edges created by the insertion of the points of $E(i,j)$ is at most:

$$c(S)\ \varepsilon_i^{-3/2} \frac{k_1}{\sqrt{\varepsilon_i}} + \sum_{k \geq 1} c(S)\ \varepsilon_i^{-3/2} \frac{k_2}{k\ c_0 \sqrt{\varepsilon_i}} = k_1\ c(S)\ \varepsilon_i^{-2} + \frac{k_2\ c(S)\ \varepsilon_i^{-2}}{c_0} \sum_{k \geq 1} \frac{1}{k}$$

The number of strips Z_k is $c'(S)\ /\ c_0 \sqrt{\varepsilon_i}$, where $c'(S)$ depends only on S. Moreover, by Theorem 2.2, the size of $E(i,j)$ is at least $c''(S)\ \varepsilon_i^{-2}$, for some constant $c''(S)$ depending only on S. It follows that n is bounded by:

$$k_1\ \frac{c(S)}{c''(S)}\ |E(i,j)| + \frac{k_2}{c_0}\ \frac{c(S)}{c''(S)}\ |E(i,j)| \sum_{1 \leq k \leq \frac{c'(S)}{c_0\ c''(S)^{1/4}} |E(i,j)|^{1/4}} \frac{1}{k}$$

i.e. $O\left(|E(i,j)|\right) + O\left(|E(i,j)|\right)\ O\left(\log |E(i,j)|\right)$. □

Finally, by subdividing the output point sample into subsets of type $E(i,j)$, with carefully chosen i, we can bound the overall number of Delaunay edges created after iteration i_0.

Theorem 5.7 *The total number of Voronoi vertices created during the course of the algorithm is $O\left(N \log N\right)$, where $N = O\left(\varepsilon^{-2}\right) = O\left(\delta^{-1}\right)$ is the size of the output point set. This bound holds for the space complexity of the algorithm.*

Proof We divide the output point set E into clusters. More precisely, i_0 is defined as above, and for any $k \geq 1$, we define i_k as the first iteration such that $r(i_k) \leq \frac{r(i_{k-1})}{8}$. Let l be the last iteration of the algorithm. We assume without loss of generality that $l = i_K$, for some K. We have $E = E(i_0) \cup \bigcup_{0 \leq k < K} E(i_k, i_{k+1})$. By Lemma 5.6, every cluster $E(i_k, i_{k+1})$ generates $|E(i_k, i_{k+1})| \log |E(i_k, i_{k+1})|$ Delaunay edges. It follows that the overall number of Delaunay edges created is at most:

$$\begin{array}{ll} & |E(i_0)|^2 + \sum_{0 \leq k < K} |E(i_k, i_{k+1})| \log |E(i_k, i_{k+1})| \\ \leq & |E(i_0)|^2 + \sum_{0 \leq k < K} |E(i_k, i_{k+1})| \log |E| \\ \leq & |E(i_0)|^2 + |E| \log |E| \end{array}$$

where $|E(i_0)| = O\left(\text{Area}(S)\ /\ \varepsilon_0^2\right)$, which depends only on S. The theorem follows, since the number of Voronoi vertices is linear with respect to the number of Delaunay edges. □

Please note that the bound given in Theorem 5.7 holds only when the surface S is fixed and the parameter ε goes to zero. As the analysis in the proof shows, the upper bound contains in fact another additive term, $N_0^2 = O\left(\varepsilon_0^{-4}\right)$, which corresponds to the size of the point sample at iteration i_0. This term, which is constant when the surface S is fixed, can be viewed as the minimum number of points needed to guarantee the topology of the output of the algorithm. As for the $O\left(N \log N\right)$ term, it dominates the other one only when N is large compared to $N_0^2 / \log N_0$, or equivalently, when ε is small compared to $\varepsilon_0^2 / \sqrt{\log 1/\varepsilon_0}$. This remark holds for the other results of Section 5 as well.

Time complexity

Lemma 5.8 *The time complexity of the algorithm is*

$$O\left(N \log^2 N\right) = O\left(\varepsilon^{-2} \log^2 \frac{1}{\varepsilon}\right) = O\left(\frac{1}{\delta} \log^2 \frac{1}{\delta}\right)$$

Proof Let T be the overall number of Delaunay tetrahedra created by the algorithm. According to Theorem 5.7, we have $T = O\left(N \log N\right)$. We will show that the time complexity is $O(T \log T)$.

- The cost of maintaining $\mathrm{Del}(E)$ is $O(T)$ since no point location is performed in our case.

- The cost of updating $\mathrm{Del}^v_{|S}(E)$ is also $O(T)$ since DETECT_ACCESS stops each time it reaches an active vertex and any vertex that becomes active remains so. Hence, the number of times a vertex is visited is at most the total number of incident Voronoi edges created by the algorithm.

- Since a Voronoi edge is probed from its vertices, it contains at most two centers of balls of $\mathrm{Del}^v_{|S}(E)$. Hence, the cost of maintaining the priority queue $\mathcal{Q}$ of bad balls of $\mathrm{Del}^v_{|S}(E)$ is $O(T \log T)$ since the total number of centers of balls of $\mathrm{Del}^v_{|S}(E)$ inserted in $\mathcal{Q}$ (and then retrieved from it) is at most twice the total number of Voronoi edges created during the process. □

5.2 Probing cost

The algorithm probes only along the Voronoi edges and from their vertices. Since every Voronoi edge has two vertices, it is probed at most twice. Hence, the total number of probes is at most twice the total number of Voronoi edges created during the process, which is $O\left(N \log N\right) = O\left(\varepsilon^{-2} \log \frac{1}{\varepsilon}\right) = O\left(\frac{1}{\delta} \log \frac{1}{\delta}\right)$, by Theorem 5.7.

5.3 Displacement cost

We bound the total number of Voronoi edges travelled by the probing device. During the update of $\mathrm{Del}^v_{|S}(E)$, two types of displacements are performed (see Section 3.1): *detection displacements* are performed inside the routine DETECT_ACCESS to locate the intersection points with the surface S; *positioning displacements* are performed during the update of $\mathrm{Del}^v_{|S}(E)$, when the probing device is moved from one place of $\mathrm{VG}(E)$ to another, before issuing a new sequence of probes.

Lemma 5.9 *The displacement cost of the algorithm is* $O\left(N^2 \log N\right) = O\left(\varepsilon^{-4} \log \frac{1}{\varepsilon}\right) = O\left(\delta^{-2} \log \frac{1}{\delta}\right)$.

Proof The overall cost of the detection displacements has been analyzed in the proof of Lemma 5.8 and shown to be $O\left(N \log N\right)$.

In addition, according to Lemma 5.3, for every iteration $i > i_0$, $E(i)$ is a $2r(i)$-sample of S, with $2r(i) \leq \varepsilon_0$. It is proved in [2] that, since $2r(i) < 0.1\ \mathrm{rch}(S)$, every Voronoi cell of $\mathrm{Vor}(E(i))$ intersects S along a topological disk that divides the cell into two components: one lies in $\mathcal{O}$, the other lies in $\mathbb{R}^3 \setminus \mathcal{O}$. Therefore, if $p(i)$ is the point inserted in E at iteration i, then, right after its insertion, all the vertices of its Voronoi cell $\mathrm{V}(p(i))$ that can be reached by the probing device will be marked *active* during the first call to DETECT_ACCESS. As a consequence, the algorithm has to call DETECT_ACCESS on only one vertex of $\mathrm{V}(p(i))$ (or on its neighbor). Hence, at iteration i, two paths only are followed by the probing device during the positioning displacements.

The lengths of these two paths are bounded by the overall number of Voronoi vertices created before iteration i. This number is $O\left(N \log N\right)$, by Theorem 5.7. Hence, the overall cost of the positioning displacements after iteration i_0 is $O\left(N.N \log N\right)$. □

The bound in Lemma 5.9 is almost tight, since on some input objects the displacement cost of the algorithm is $\Omega\left(N^2\right)$. Figure 4 presents an example in the plane. The top image shows the object $\mathcal{O}$ and the initial point sample E_{init}, both symmetric with respect to the origin (marked by a point at the center of the object). The bottom image shows the point sample E and the balls of $\text{Del}^v_{|S}(E)$ at some stage of the course of the algorithm. Since at each iteration the algorithm inserts the center of the largest ball of $\text{Del}_{|S}(E)$, it is easily seen that E remains symmetric (or almost symmetric) with respect to the origin throughout the process. Hence, each time a point p lying inside a cavity is inserted in E, the iteration before or after the algorithm inserts in E the symmetric of p, which lies in the other cavity. Since the density of the output point sample is uniform, the number of points inserted inside the cavities (and hence also the number of Voronoi edges lying in the cavities) is linear with respect to N. Therefore, the overall number of Voronoi edges travelled by the probing device is $\Omega\left(N^2\right)$.

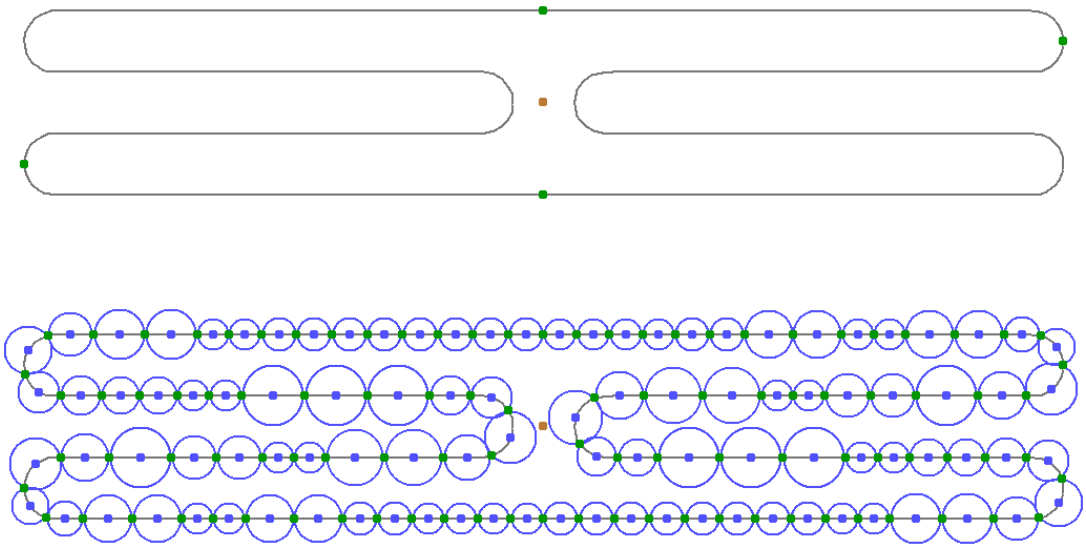

Figure 4: A quadratic example.

6 Dealing with more than one connected component

Let $S_1, \cdots, S_n$ be the connected components of S. We assume that these components are not nested, which is no real loss of generality since, otherwise, the probing device would not be able to probe all the components. Under this assumption, $\mathbb{R}^3 \setminus \mathcal{O}$ is path-connected, and $\mathcal{O}$ has n connected components exactly, $\mathcal{O}_1, \cdots, \mathcal{O}_n$, such that $\mathcal{O}_i$ is bounded by S_i, for all i. According to A1, for every component $\mathcal{O}_i$ we are given a point $o_i \in \mathcal{O}_i$. We assume that $\varepsilon < 0.05\, \varepsilon_S$.

To mesh the surface, we build a persistent facet on some component of S and we run the algorithm. Upon termination, we are able to check which components of S have been meshed. Therefore, we iterate the process, building a persistent facet on an unmeshed component and running the algorithm again, until all the connected components of S are meshed. We will now

review this procedure in details. The validity of the approach relies on several lemmas, whose proofs have been added for completeness but can be skipped in a first reading.

6.1 Meshing one component of S

For the initialization, we choose some radom position p on $\partial\Omega$ and we probe towards o_1. The construction of a persistent facet (a, b, c) is done by the same means as in Section 3.2. Note however that (a, b, c) may not lie on S_1, since the latter may be hidden from p by another connected component of S.

After this initialization step, we run the algorithm of Section 3. Upon termination, Theorem 2.1 holds with S replaced by the union U of the connected components of S that contain vertices of $\mathrm{Del}^v_{|S}(E)$. We claim that U is not empty. Indeed, as a persistent facet, (a, b, c) remains in $\mathrm{Del}_{|S}(E)$ throughout the process, which implies that $\mathrm{VG}(E) \cap S$ is not empty upon termination. Since $\mathrm{VG}(E)$ is a connected graph, at least one point of $\mathrm{VG}(E) \cap S$ can be reached from an infinite Voronoi vertex by travelling along free Voronoi edges. Hence, $\mathrm{Del}^v_{|S}(E)$ has at least one facet[6] and $U \neq \emptyset$. Moreover,

Lemma 6.1 $\mathrm{Del}^v_{|S}(E)$ *is equal to* $\mathrm{Del}_{|S}(E)$.

Proof By L3, any ball B of $\mathrm{Del}^v_{|S}(E)$ has a connected intersection with S, hence its center lies on the same connected component of S as the vertices of the facet of $\mathrm{Del}^v_{|S}(E)$ circumscribed by B. It follows that U is the only part of S that contains centers of balls of $\mathrm{Del}^v_{|S}(E)$. As a consequence, $\mathrm{Del}^v_{|S}(E)$ is equal to $\mathrm{Del}^v_{|U}(E)$, the visible Delaunay triangulation of E restricted to U.

Since $\varepsilon < 0.05\ \varepsilon_S$, E is a 0.1-sample of U, by Theorem 2.1. It follows that the Delaunay triangulation of E restricted to U, $\mathrm{Del}_{|U}(E)$, is homeomorphic to U, by Theorem 2 of [2]. Hence, $\mathrm{Del}^v_{|U}(E)$ and $\mathrm{Del}_{|U}(E)$ are equal, since they are homeomorphic and the former is a subcomplex of the latter[7].

To conclude the proof of the lemma, it suffices to prove that $\mathrm{Del}_{|U}(E) = \mathrm{Del}_{|S}(E)$. Let us assume the contrary. Then, $\mathrm{VG}(E)$ intersects $S \setminus U$. Let c be a point of $\mathrm{VG}(E) \cap (S \setminus U)$. Since $\mathrm{VG}(E)$ is a connected graph, there exists a connected path π inside $\mathrm{VG}(E)$ that goes from c to an infinite Voronoi vertex v. Let w be the first Voronoi vertex of π.
- If $\pi \setminus \{c\}$ does not intersect S, then w belongs to the same connected component of $\mathrm{VG}(E) \setminus S$ as v. Hence, by P2, v and w have the same status, which is active since v is infinite. It follows that c is the center of a ball of $\mathrm{Del}^v_{|S}(E)$, which means that S_i belongs to U, which contradicts our assumption.
- If $\pi \setminus \{c\}$ intersects S, then we can assume without loss of generality that $\pi \setminus \{c\}$ does not intersect $S \setminus U$. Otherwise, it suffices to take for c the last point of $S \setminus U$ on π. Let c' be the first point of S on $\pi \setminus \{c\}$. As assumed above, c' belongs to U and is therefore the center of a ball of $\mathrm{Del}_{|U}(E)$. Moreover, since $c \in S \setminus U$ and since the connected components of S are not nested, the arc $]c, c'[$ of π lies outside the object $\mathcal{O}$. Let e' be the Voronoi edge that contains c'. Since E is a 0.1-sample of U, e' intersects S only at c', by L2. Hence, one of its vertices belongs to $\mathcal{O}$ (and is thus inactive), while its other vertex (say z) lies outside $\mathcal{O}$ (and hence belongs to the arc $]c, c'[$ of π). Now, since $\mathrm{Del}^v_{|U}(E) = \mathrm{Del}_{|U}(E)$, one of the vertices of e' is active. This vertex must be z because the other

[6] See the proof of H2 (Section 4.3) for a similar argument.
[7] The same argument is invoked at the very end of Section 4.3.

vertex is in $\mathcal{O}$. Therefore, one vertex of the arc $]c, c'[$ of π is active, which by P2 implies that all the vertices of the arc are active, and among them w. It follows that c is the center of a ball of $\mathrm{Del}^v_{|S}(E)$, which means that $c \in U$, hereby contradicting our assumption. $\square$

6.2 Meshing the other components of S

To mesh the other connected components of S, we must first determine which components have been meshed so far. Let $\mathcal{O}^U$ be the union of the components of $\mathcal{O}$ whose boundaries belong to U, and let $\mathcal{O}^v$ be the bounded open set of $\mathbb{R}^3$ whose boundary is $\mathrm{Del}^v_{|S}(E)$. Since U has no nested connected components, $\mathbb{R}^3 \setminus \mathcal{O}^U$ is path-connected. Moreover, since U and $\mathrm{Del}^v_{|S}(E)$ are ambient isotopic, $\mathbb{R}^3 \setminus \mathcal{O}^v$ is also path-connected.

Lemma 6.2 *The Hausdorff distance betwen $\mathcal{O}^U$ and $\mathcal{O}^v$ is at most ε.*

Proof Let $\mathrm{dist_H}(A, B)$ stand for the Hausdorff distance between two sets A and B. We know from Theorem 2.1 that $\mathrm{dist_H}(U, \mathrm{Del}^v_{|S}(E)) \leq \varepsilon$. However, this fact alone does not imply that $\mathrm{dist_H}(\mathcal{O}^U, \mathcal{O}^v) \leq \varepsilon$. Let $\mathcal{T}_\varepsilon$ be the so-called *tubular neighborhood* of U of width ε, *i.e.* the set of the points of $\mathbb{R}^3$ whose distance to U is at most ε. We call U_- the part of the boundary of $\mathcal{T}_\varepsilon$ that lies in $\mathcal{O}^U$, and U_+ the other part of the boundary of $\mathcal{T}_\varepsilon$ (which lies in $\mathbb{R}^3 \setminus \mathcal{O}^U$). It is a well-known result of differential topology [24, Ch. 5] that U_- and U_+ are ambient isotopic to U. Let $\mathcal{O}^U_-$ and $\mathcal{O}^U_+$ be the bounded open sets of $\mathbb{R}^3$ whose boundaries are respectively U_- and U_+. It is easily seen that $\mathcal{O}^U_- = \mathcal{O}^U \setminus \mathcal{T}_\varepsilon$ and $\mathcal{O}^U_+ = \mathcal{O}^U \cup \mathcal{T}_\varepsilon$, which implies that $\mathrm{dist_H}(\mathcal{O}^U, \mathcal{O}^U_-)$ and $\mathrm{dist_H}(\mathcal{O}^U, \mathcal{O}^U_+)$ are bounded by ε. We will show that the following relation holds, which will prove the lemma:

$$\mathcal{O}^U_- \subseteq \mathcal{O}^v \subseteq \mathcal{O}^U_+ \tag{1}$$

Let $p \in \mathbb{R}^3 \setminus \mathcal{O}^U_+$. Since $\mathbb{R}^3 \setminus \mathcal{O}^U$ is path-connected and U_+ is ambient isotopic to U, $\mathbb{R}^3 \setminus \mathcal{O}^U_+$ is path-connected, which implies that there is a path π from p to infinity that does not intersect $\mathcal{O}^U_+$. Now, $\mathrm{Del}^v_{|S}(E)$ is included in the union of the balls of $\mathrm{Del}^v_{|S}(E)$, which is contained in $\mathcal{T}_\varepsilon \subseteq \mathcal{O}^U_+$. Therefore, π does not intersect $\mathrm{Del}^v_{|S}(E)$ either, which means that $p \in \mathbb{R}^3 \setminus \mathcal{O}^v$ because $\mathcal{O}^v$ is bounded. Hence, $\mathbb{R}^3 \setminus \mathcal{O}^U_+ \subseteq \mathbb{R}^3 \setminus \mathcal{O}^v$, or equivalently, $\mathcal{O}^v \subseteq \mathcal{O}^U_+$, which proves the right-hand side of (1).

Let now p be a point of $\mathcal{O}^U_-$. We build a path $\pi(p)$ from p to infinity as follows:

- Let r be any ray issued from p. We call p_- the first point of U_- crossed by r.
- Let $\tilde{p}$ be the point of U closest to p_-. Since p_- stands on $U_- \subset \mathcal{T}_\varepsilon$, $\tilde{p}$ is unique, and the line $(p_-, \tilde{p})$ is aligned with the normal of S at $\tilde{p}$. We call *fiber of* $\tilde{p}$, or simply $\mathrm{Fib}(\tilde{p})$, the segment of $(p_-, \tilde{p}) \cap \mathcal{T}_\varepsilon$ that contains $\tilde{p}$. One endpoint of $\mathrm{Fib}(\tilde{p})$ is p_- and lies on U_-, the other endpoint (say p_+) lies on U_+.
- Since $p_+ \in U_+$, there is a path $\pi_+ \subset \mathbb{R}^3 \setminus \mathcal{O}^U_+$ that connects p_+ to infinity.
- Finally, we define $\pi(p)$ as follows:

$$\pi(p) = [p, p_-[\cup [p_-, p_+] \cup \pi_+$$

Note first that $[p, p_-[$ and π_+ do not intersect $\mathrm{Del}^v_{|S}(E)$, since the latter is included in $\mathcal{T}_\varepsilon$ while $[p, p_-[$ and π_+ are not. As a result, p_+ lies in $\mathbb{R}^3 \setminus \mathcal{O}^v$, while p and p_- lie on the same side of $\mathrm{Del}^v_{|S}(E)$ (either $\mathcal{O}^v$ or $\mathbb{R}^3 \setminus \mathcal{O}^v$). Moreover, the proof of isotopy between U and $\mathrm{Del}^v_{|S}(E)$ provided in [7] (which in fact refers to the proof given in [3]), states that every fiber of U intersects $\mathrm{Del}^v_{|S}(E)$ exactly once, and then uses this fact to work out the isotopy. Therefore, $[p_-, p_+] = \mathrm{Fib}(\tilde{p})$ intersects $\mathrm{Del}^v_{|S}(E)$ in exactly one point $\hat{p}$. We can assume without loss of generality that $\hat{p}$ belongs to the relative interior of a facet f of $\mathrm{Del}^v_{|S}(E)$, since it is always possible to move p_- slightly so as to ensure this property. The intersection of $\mathrm{Fib}(\tilde{p})$ with f is then transversal, which means that p_- and p_+ do not lie on the same side of $\mathrm{Del}^v_{|S}(E)$. Since $p_+ \in \mathbb{R}^3 \setminus \mathcal{O}^v$, p_- (and hence p) lies in $\mathcal{O}^v$. Since this is true for any point $p \in \mathcal{O}^U_-$, the left-hand side of (1) is proved. □

To see which components of S belong to U, we determine, for every $\mathcal{O}_i$, whether o_i satisfies any of the following conditions:

C1 $\mathrm{dist}(o_i, E) < \varepsilon_S$

C2 $o_i \in \mathcal{O}^v$

Lemma 6.3 *$S_i \subseteq U$ if, and only if, o_i satisfies (C1) or (C2).*

Proof Assume first that $S_i \subseteq U$. Then, $o_i \in \mathcal{O}_i \subseteq \mathcal{O}^U$. If $o_i \in \mathcal{O}^v$, then (C2) is satisfied. Otherwise, by Lemma 6.2, the distance from o_i to $\partial\mathcal{O}^v = \mathrm{Del}^v_{|S}(E)$ is at most ε. Let $p \in \mathrm{Del}^v_{|S}(E)$ be a nearest neighbor of o_i, and $q \in E$ a nearest neighbor of p. Since the facets of $\mathrm{Del}^v_{|S}(E)$ have circumradii of at most ε, $\mathrm{dist}(p, q)$ is bounded by ε, hence

$$\mathrm{dist}(o_i, E) \leq \mathrm{dist}(o_i, q) \leq \mathrm{dist}(o_i, p) + \mathrm{dist}(p, q) \leq 2\varepsilon < \varepsilon_S,$$

which means that (C1) is satisfied.

Assume now conversely that S_i does not belong to U. Then $\mathrm{dist}(o_i, \mathcal{O}^U) \geq 2\ \varepsilon_S$, because the components of $\mathcal{O}$ are farther than $2\ \mathrm{rch}(S) > 2\ \varepsilon_S$ from one another. This implies that $\mathrm{dist}(o_i, E) \geq 2\ \varepsilon_S$ since $E \subset U$. Hence, (C1) is not satisfied. Moreover, by Lemma 6.2, we have

$$\mathrm{dist}(o_i, \mathcal{O}^v) \geq \mathrm{dist}(o_i, \mathcal{O}^U) - \mathrm{dist}_{\mathrm{H}}(\mathcal{O}^U, \mathcal{O}^v) \geq 2\ \varepsilon_S - \varepsilon > 0,$$

which means that o_i does not belong to $\mathcal{O}^v$, and thus that (C2) is not satisfied either. □

Thanks to Lemma 6.3, we know precisely which connected components of S have been meshed, by checking which of the o_i satisfy (C1) or (C2).

- Checking (C1) with o_i reduces to finding the point of E that is closest to o_i, which can be performed by locating o_i in $\mathrm{Vor}(E)$.
- Regarding (C2), we notice that every Delaunay tetrahedron lies either completely inside $\mathcal{O}^v$ or completely outside $\mathcal{O}^v$, since the facets of $\mathrm{Del}^v_{|S}(E)$ belong to the Delaunay triangulation. Hence, it suffices to mark each tetrahedron as *interior* or *exterior*, and then to locate each o_i in $\mathrm{Del}(E)$.

As a consequence, (C1) and (C2) can be checked for all the o_i in $O(n \log |E| + T)$ time, where n is the number of connected components of S and T is the number of tetrahedra of $\mathrm{Del}(E)$.

Once we have determined which connected components of S remain to mesh, we want to create a persistent facet (a', b', c') on $S \setminus U$. This requires to be able to probe points of $S \setminus U$.

Lemma 6.4 *We can work out a point of $\mathbb{R}^3 \setminus \mathcal{O}$, reachable by the probing device, from which it is possible to probe points of $S \setminus U$.*

Proof As explained above, we know precisely which connected components of S belong to U. Hence, we know an i such that $S_i \cap U = \emptyset$. Let $p \in E$ be the point of E that is closest to o_i. By definition, the cell $\mathrm{V}(p)$ of p in $\mathrm{Vor}(E)$ contains o_i. It follows that the cells $V^+(p)$ and $V^+(o_i)$ in $\mathrm{Vor}(E \cup \{o_i\})$ have a non-empty intersection. Note that the edges of $\partial V(p)$ do not intersect S_i, since otherwise Lemma 6.1 would imply that S_i contains the center of a ball of $\mathrm{Del}^v_{|S}(E)$ and hence belongs to U, which contradicts our assumption. Moreover, $V^+(o_i)$ does not intersect U. Indeed, for any point q of $V^+(o_i)$, we have $\mathrm{dist}(q, E) \geq \frac{1}{2}\,\mathrm{dist}(o_i, E)$, which is greater than ε_S because $E \subset U$ and $o_i \in S \setminus U$. Now, E is a 2ε-sample of U, thus no point of U is farther than $2\varepsilon \leq \varepsilon_S$ from E, which means that $q \notin U$.

We compute the edges and vertices of the boundary of $V^+(o_i)$, and then we move the probing device along the free edges of the boundary of $\mathrm{V}(p)$, starting from an active vertex (which exists because p is incident to a facet of $\mathrm{Del}^v_{|S}(E)$), until we reach a vertex of $V^+(o_i)$.

It is proved in [2] that, since E is a 2ε-sample of U, with $2\varepsilon < 0.1\ \varepsilon_S \leq 0.1\ \mathrm{rch}(S)$, $\mathrm{V}(p)$ intersects U along a topological disk that divides $\mathrm{V}(p)$ into two components: one lies in $\mathcal{O}^U$, the other lies in $\mathbb{R}^3 \setminus \mathcal{O}^U$. Since the edges of $\partial\mathrm{V}(p)$ do not intersect $S \setminus U$, we will eventually find a vertex v of $V^+(o_i)$ by following the free edges of $\partial\mathrm{V}(p)$.

Once the probing device is at v, we probe from v towards o_i and find some point $s \in S$. Since o_i and v both lie in $V^+(o_i)$, which is convex, the segment $[v, o_i]$ is included in $V^+(o_i)$. Hence, $s \in V^+(o_i)$. Since $V^+(o_i)$ does not intersect U, $s \in S \setminus U$, which ends the proof of the lemma. □

Once we have built a persistent facet (a', b', c') on $S \setminus U$ using the method of Section 3.2, we run the algorithm with $E \cup \{a', b', c'\}$ as input point sample. Upon termination, (a', b', c') is still a facet of $\mathrm{Del}_{|S}(E)$, by Lemma 7.1 of [7]. Hence, the connected component of S that contains (a', b', c') belongs to U, by Lemma 6.1.

As explained at the beginning of Section 6, we iterate this process of creating persistent facets on unmeshed components of S and running the algorithm again, until all the connected components of S are meshed, which will eventually happen. At this stage, all the o_i satisfy (C1) or (C2), thus we know that we can stop. Observe that the algorithm is run at most n times, where n is the number of connected components of S.

7 Implementation and results

We have implemented the algorithm using the CGAL library [38] which provided us with robust and flexible implementations of the Delaunay triangulation in 2D and in 3D. A video [6] is available online, which describes the algorithm and demonstrates its practicality. Results on a planar curve and on a surface are reported in Figures 5 and 6. In the electronic version of the paper, the active part of the Voronoi graph is printed in blue, the inactive part in black, and the faces of $\mathrm{Del}^v_{|S}(E)$ are shown in red or in green, depending on whether they are circumscribed by a good or a bad ball of $\mathrm{Del}^v_{|S}(E)$. In the 2D example, the inactive part of the Voronoi graph is shown only in the first image, for clarity.

Please note that the examples shown in the paper result from simulations on implicit surfaces, and that the method has not yet been tested on a real physical system. As experimental results show, our probing algorithm can be used as a surface mesher, provided that the oracle can be

implemented – which is the case for implicit surfaces. Nevertheless, the algorithm is inherently less efficient than its predecessor, due mainly to the displacement cost which does not exist in Chew's algorithm.

8 Conclusion

Many important questions are left open by this work, including:

- Can the number of probes be reduced to $O(N)$, where N is the size of the output point sample? As emphasized in [4], in the case where $\mathcal{O}$ is a set of pairwise disjoint convex sets, the number of Voronoi vertices created outside $\mathcal{O}$ is linear with respect to N, hence the number of probes is $O(N)$.
- A trivial upper bound on the total Euclidean distance travelled by the probing device is $O(\Delta\, N^2 \log N)$, where Δ is the diameter of the compact convex set Ω containing S. However, this bound is too coarse since most Voronoi edges are short (and close to the medial axis). Can a tighter bound be worked out?
- What are the exact trade-offs between optimizing the combinatorial cost and the displacement cost?
- Our algorithm is certified provided that the user-defined parameter ε is sufficiently small compared to $\mathrm{rch}(S)$. One way to ensure this condition is to know a positive lower bound ε_S on $\mathrm{rch}(S)$, and to choose ε less than a fraction of ε_S, as assumed in the paper. However, such a lower bound is not readily available in all practical situations. Therefore, it would be interesting to see if, using a stronger probe model, one can devise an algorithm that does not rely on this assumption. The methods of [12, 30] might be good candidates.
- We have assumed a perfect probing device. How can we model uncertainty in the probes? Some related results can be found in [19, 20].
- Can the approach be extended to piecewise smooth surfaces?
- In practice a physical scaffold has to be present around the object being sampled to support the probing device. Can we show similar results for a probing device whose motions obey realistic constraints?
- Can we extend the approach to more general manifolds in higher dimensions? In particular, can we adapt our probe model so that it holds for higher codimensions? Can we avoid computing the full dimensional Delaunay triangulation, whose cost becomes prohibitive?

9 Acknowledgements

We thank the anonymous referees for their insightful comments and their numerous suggestions to improve the quality of the paper. The first and third authors were partially supported by the European Union through the Network of Excellence AIM@SHAPE Contract IST 506766. The second author was supported in part by NSF CARGO grant 0138456, NSF ITR grant 0205671 and NSF FRG grant 0354543.

References

[1] P. D. Alevizos, J.-D. Boissonnat, and M. Yvinec. Non-convex contour reconstruction. *J. Symbolic Comput.*, 10:225–252, 1990.

[2] N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *Discrete Comput. Geom.*, 22(4):481–504, 1999.

[3] N. Amenta, S. Choi, T. K. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. *Internat. Journal of Comput. Geom. and Applications*, 12:125–141, 2002.

[4] D. Attali, J.-D. Boissonnat, and A. Lieutier. Complexity of the Delaunay triangulation of points on surfaces: the smooth case. In *Proc. 19th Annu. ACM Sympos. Comput. Geom.*, pages 201–210, 2003.

[5] J.-D. Boissonnat, D. Cohen-Steiner, and G. Vegter. Isotopic implicit surface meshing. In *Proc. 36th Annu. ACM Sympos. on Theory of Computing*, pages 301–309, 2004.

[6] J.-D. Boissonnat, L. J. Guibas, and S. Oudot. Learning smooth objects by probing. In *Proc. 21st Annu. ACM Sympos. Comput. Geom.*, pages 364–365, 2005. Video available at: `http://compgeom.poly.edu/acmvideos/socg05video/index.html`.

[7] J.-D. Boissonnat and S. Oudot. Provably good sampling and meshing of surfaces. *Graphical Models*, 67(5):405–451, September 2005.

[8] J.-D. Boissonnat and Mariette Yvinec. Probing a scene of non-convex polyhedra. *Algorithmica*, 8:321–342, 1992.

[9] M. Do Carmo. *Riemannian Geometry*. Birkhäuser, Boston, Basel, Berlin, 1992.

[10] H.-L. Cheng, T. K. Dey, H. Edelsbrunner, and J. M. Sullivan. Dynamic skin triangulation. *Discrete and Computational Geometry*, 25:525–568, 2001.

[11] H.-L. Cheng and X. Shi. Guaranteed quality triangulation of molecular skin surfaces. In *VIS '04: Proceedings of the conference on Visualization '04*, pages 481–488, Washington, DC, USA, 2004. IEEE Computer Society.

[12] S.-W. Cheng, T. K. Dey, E. A. Ramos, and T. Ray. Sampling and meshing a surface with guaranteed topology and geometry. In *Proc. 20th Annu. ACM Sympos. on Computat. Geom.*, pages 280–289, 2004.

[13] E. Chernyaev. Marching cubes 33: Construction of topologically correct isosurfaces. Technical report CN 95-17, CERN, 1995.

[14] L. P. Chew. Guaranteed-quality mesh generation for curved surfaces. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 274–280, 1993.

[15] Howie Choset and Joel Burdick. Sensor-based exploration: The hierarchical generalized Voronoi graph. *The International Journal of Robotics Research*, 19:96–125, 2000.

[16] Howie Choset, Sean Walker, Kunnayut Eiamsa-Ard, and Joel Burdick. Sensor-based exploration: Incremental construction of the hierarchical generalized Voronoi graph. *The International Journal of Robotics Research*, 19:126–148, 2000.

[17] K. L. Clarkson. Building triangulations using ε-nets. In *Proc. 38th Annu. Sympos. on Theory of Computing*, 2006.

[18] R. Cole and C. K. Yap. Shape from probing. *J. Algorithms*, 8(1):19–38, March 1987.

[19] T. K. Dey and S. Goswami. Provable surface reconstruction from noisy samples. In *Proc. 20th Annu. ACM Sympos. Comput. Geom.*, pages 330–339, 2004.

[20] D. P. Dobkin, H. Edelsbrunner, and C. K. Yap. Probing convex polytopes. In *Proc. 18th Annu. ACM Sympos. Theory Comput.*, pages 424–432, 1986.

[21] H. Edelsbrunner and N. R. Shah. Triangulating topological spaces. *Int. J. on Comp. Geom.*, 7:365–378, 1997.

[22] H. Federer. *Geometric Measure Theory*. Classics in Mathematics. Springer-Verlag, 1996. Reprint of the 1969 ed.

[23] J. C. Hart and B. T. Stander. Guaranteeing the topology of an implicit surface polygonization for interactive modeling. In *Proc. SIGGRAPH*, pages 279–286, 1997.

[24] M. W. Hirsch. *Differential Topology*. Springer-Verlag, New York, NY, 1976.

[25] Nico Kruithof and Gert Vegter. Meshing skin surfaces with certified topology. Technical Report ECG-TR-364100-02, Rijksuniversiteit Groningen, 2004.

[26] P. Laug and H. Borouchaki. Molecular surface modeling and meshing. In *Proc. 10th International Meshing Roundtable*, pages 31–41, 2001.

[27] M. Lindenbaum and A. M. Bruckstein. Blind approximation of planar convex sets. *IEEE Trans. Robot. Autom.*, 10(4):517–529, August 1994.

[28] W. Lorensen and H. Cline. Marching cubes: a high resolution 3d surface construction algorithm. *Comput. Graph.*, 21(4):163–170, 1987.

[29] S. Oudot. *Sampling and Meshing Surfaces with Guarantees.* Thèse de doctorat en sciences, École Polytechnique, Palaiseau, France, 2005. Preprint available at `http://geometry.stanford.edu/member/oudot/publis/oudot_dissertation.pdf`.

[30] S. Plantinga and G. Vegter. Isotopic approximation of implicit curves and surfaces. In *Proc. 2nd Sympos. on Geometry Processing*, pages 251–260, 2004.

[31] T. J. Richardson. Approximation of planar convex sets from hyperplanes probes. *Discrete and Computational Geometry*, 18:151–177, 1997.

[32] G. Rote. The convergence rate of the Sandwich algorithm for approximating convex functions. *Computing*, 48:337–361, 1992.

[33] K. Shimada and D. C. Gossard. Automatic triangular mesh generation of trimmed parametric surfaces for finite element analysis. *Comput. Aided Geom. Des.*, 15(3), 1998.

[34] S. S. Skiena. Geometric reconstruction problems. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 26, pages 481–490. CRC Press LLC, Boca Raton, FL, 1997.

[35] J. F. Traub, G. W. Wasilkowski, and H. Wozniakowski. *Information-based Complexity.* Academic Press, 1988.

[36] J. F. Traub and A. G. Werschulz. *Complexity and Information.* Cambridge University Press, 1998.

[37] J. R. Tristano, S. J. Owen, and S. A. Canann. Advancing front surface mesh generation in parametric space using a Riemannian surface definition. In *Proc. 7th international meshing reoundtable*, pages 429–445, 1998.

[38] `http://www.cgal.org`.

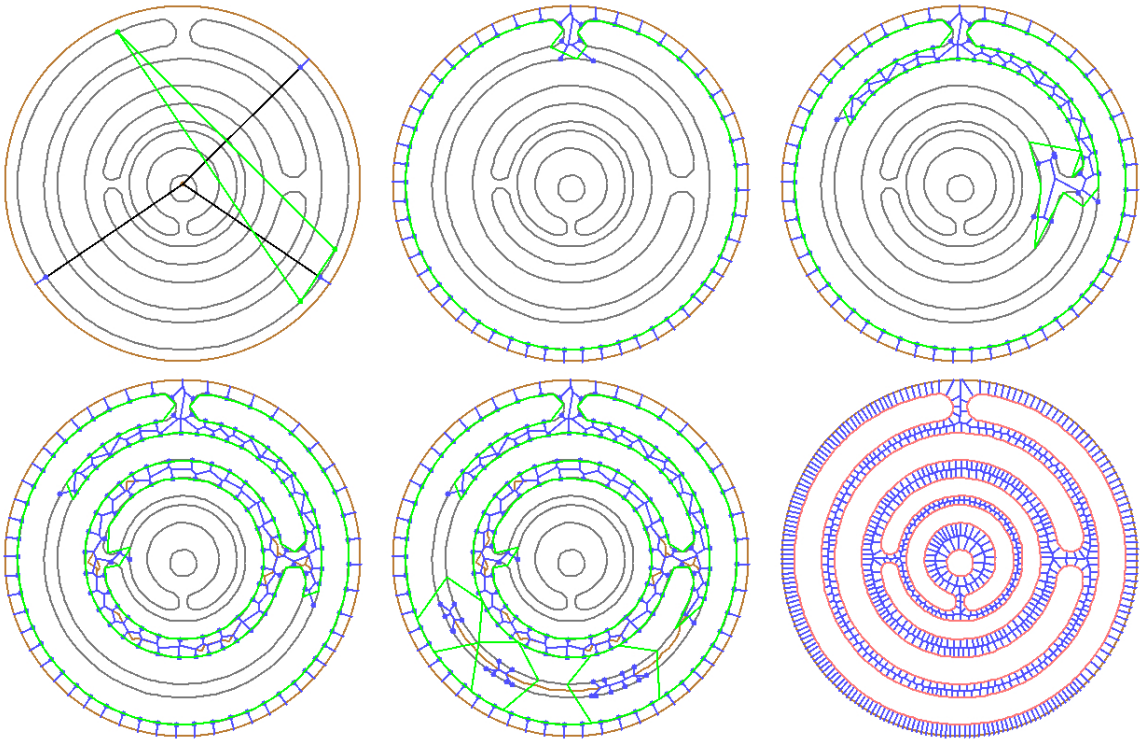

Figure 5: Course of the algorithm on a curve in $\mathbb{R}^2$

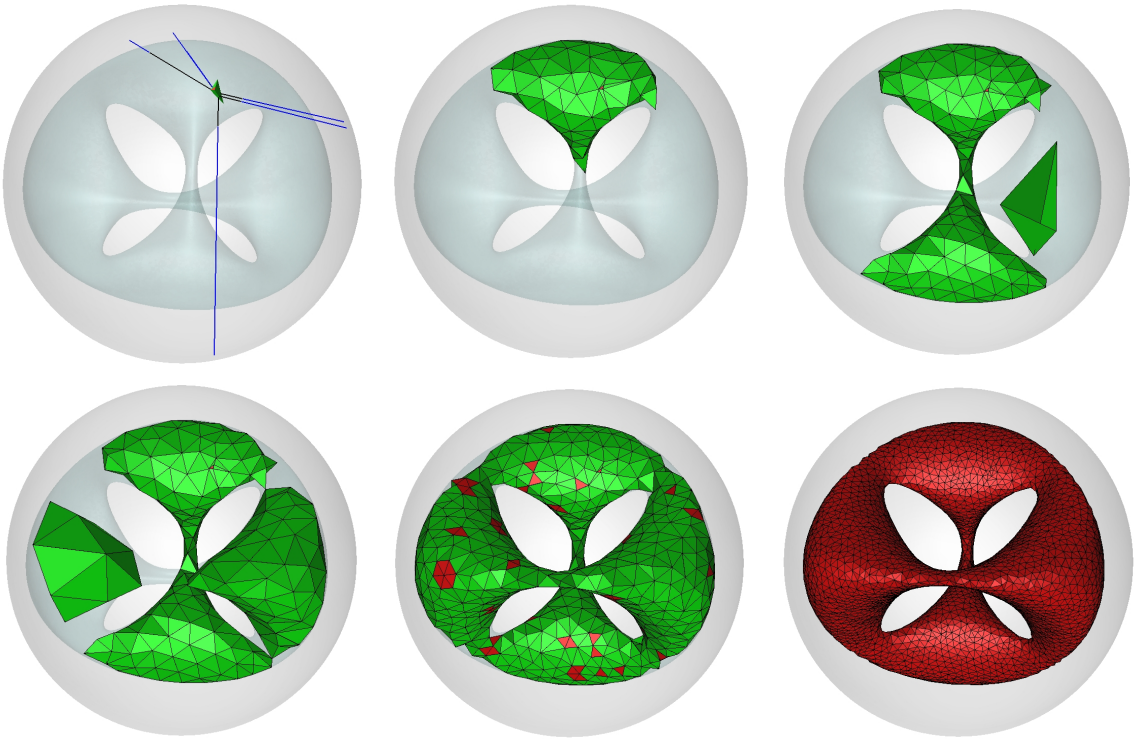

Figure 6: Course of the algorithm on a surface in $\mathbb{R}^3$