Perturbations and Vertex Removal in a 3D Delaunay Triangulation

Olivier Devillers, Monique Teillaud

Perturbations and Vertex Removal in a 3D Delaunay Triangulation

Olivier Devillers* Monique Teillaud*

Abstract

Though Delaunay triangulations are very well known geometric data structures, the problem of the robust removal of a vertex in a three-dimensional Delaunay triangulation is still a problem in practice.
We propose a simple method that allows to remove any vertex even when the points are in very degenerate configurations. The solution is available in CGAL[1].

1 Introduction

Let us first briefly recall that the Delaunay triangulation of a set $\mathcal{S}$ of points in $\mathbf{R}^3$ is the partition of the convex hull of the points consisting of the tetrahedra whose circumscribing ball does not contain any point of $\mathcal{S}$ in its interior.
This definition implicitly implies that the tetrahedra are not flat, otherwise their circumscribing ball is not defined. In fact, it is always possible to build a Delaunay triangulation of a given set of points without any flat tetrahedron, as will be noticed in § 4.2.
When a vertex v is removed from the Delaunay triangulation $\mathcal{DT}(\mathcal{S})$ of $\mathcal{S}$, the tetrahedra incident to v are removed, which creates a polyhedral hole, and the interior of this polyhedron H must be triangulated with Delaunay tetrahedra. Triangulating H is exactly the inverse operation of inserting v in $\mathcal{DT}(\mathcal{S} \setminus \{v\})$. After the removal, the Delaunay triangulation is the triangulation that would have been obtained if v had never been inserted.
When degeneracies occur (at least five cospherical points in $\mathcal{S}$), the Delaunay triangulation is not unique. In general, any of the possible triangulations of the set of cospherical points can be returned.
The difficulty arises when there are at least four cocircular points p_1, p_2, p_3, p_4, i.e. points that are both cospherical and coplanar, on the boundary of H. Indeed, in this case, for any point p_5 on H, the five points p_1, p_2, p_3, p_4, p_5 are cospherical, and there are two possible triangulations of this set of points, corresponding to the two different choices for the diagonal of the convex polygon p_1, p_2, p_3, p_4 which is a facet of H (or a subfacet in the case when there are more cocircular points). Depending on this choice, we will get a different triangulation of this facet. In fact, there is no choice: since the outside of H is already triangulated, the triangulations of all the facets of H are given and must be respected.

Note that a similar problem cannot occur in the planar case, since the hole created by the removal of a vertex is a polygon, so, even in the case when this polygon has collinear vertices, there is no choice to be made: the edges of the polygon are uniquely determined by the order of the collinear vertices on the line that contains them. Any triangulation of the interior of the polygon will have the same edges on the polygon.

The problem can be seen as a special instance of the following question:

(1.1) *Is it always possible to compute a Delaunay triangulation of a given polyhedron H in such a way that the triangulation of its facets is respected?*

In this paper, we show that, though this general question has a negative answer, the more restrictive instance of the problem posed by the vertex removal from a 3D Delaunay triangulation constructed incrementally can be solved, even in degenerate situations, and we propose a simple method to achieve it.

2 Previous Work

2.1 Algorithms. The deletion of vertices in two dimensional triangulations has been widely studied from both theoretical and practical points of view. A degree k vertex v can be removed in optimal $\Theta(k)$ time [1], [5], but in practice people often use sub-optimal solutions in $O(k^2)$ or $O(k \log k)$ time [6].

In higher dimensions, the problem has received less attention since the complexity of the result can reach $\Omega(k^2)$. Thus, a worst-case optimal solution consists in

*INRIA - BP 93 - 06902 Sophia Antipolis cedex - France
{Olivier.Devillers, Monique.Teillaud}@sophia.inria.fr
http://www-sop.inria.fr/prisme/personnel/devillers/
http://www-sop.inria.fr/galaad/teillaud/

[1] http://www.cgal.org (releases 2.3 and 2.4)

computing the Delaunay triangulation of the neighbors of v in $\mathcal{DT}(\mathcal{S})$ (i.e. the vertices of H), then removing from this small triangulation the tetrahedra that are outside H and finally *"sewing"* this triangulation in the hole of $\mathcal{DT}(\mathcal{S})$ created by the deletion of v. Though this approach leads to a worst-case asymptotic optimal algorithm, it must be noticed that not only the hole is triangulated, but the whole convex hull of this hole, so, many tetrahedra are constructed and then deleted, which is useless memory management work.

To avoid this sewing technique and triangulate only the hole created by the deletion of v, another approach is the generalization of the 2D *ear algorithm* [13], [6]. The hole H is a simple polyhedron; the general idea of the algorithm is to add new tetrahedra one by one in a way that ensures that the hole stays simple; such a tetrahedron is called an *ear*. There are two kinds of ears: ears formed by two triangles on the boundary of the hole, that share an edge, and ears formed by three triangles incident to a vertex that has degree 3 on the boundary of the hole.
At each step, there may be several ears satisfying the Delaunay criterion; Devillers proves that the ear for which the removed point has the smallest power with respect to the circumscribing sphere belongs to the Delaunay triangulation [6], and he proposes to choose this ear. By maintaining a priority queue of the candidate ears, this yields a theoretical complexity of $O(f \log k)$, where f is the number of tetrahedra to be created.

2.2 Perturbation Techniques. A general approach to solve the degenerate cases is the use of symbolic perturbations [16]. The general idea is to make the problem dependent of a parameter ε such that:

- there exists $\varepsilon_0 > 0$ such that the parameterized problem is in general position for $\varepsilon \in (0, \varepsilon_0]$,
- if the original problem is in general position, the solution of the parameterized problem tends to the solution of the original problem when ε goes to zero,
- if the original problem is not in general position, the solution of the parameterized problem tends to something relevant when ε goes to zero by positive values.

More precisely, if $\mathcal{S}$ is not in general position, the Delaunay triangulation is not uniquely defined and the aim of a perturbation technique is to select one of the possible Delaunay triangulations.

Different perturbation methods can be used in that case: either the input of the problem can be perturbed or the definition of the problem can be slightly changed.

When using a perturbation, some properties can be lost in the result: to check whether they are satisfied, we must
- check that the parameterized solution satisfies the properties
- check that the properties are still true at the limit.
Perturbing the input can have very serious drawbacks: if the points move with ε [2], then a non flat tetrahedron can become flat at the limit. By *flat* tetrahedron, we mean a tetrahedron whose four vertices are coplanar. As explained in § 4, we want to avoid this.

A standard way of changing the problem definition is to add to each point a weight depending on ε which goes to zero when ε goes to zero and to compute the regular triangulation (defined as the dual of the power diagram) instead of the Delaunay triangulation [10].
One important advantage is that the points do not move, thus if a tetrahedron belongs to the limit solution, the same tetrahedron exists in the solution of a non degenerate problem and thus this tetrahedron is non flat. But the drawback of this technique is that some properties of Delaunay triangulations satisfied for non-degenerate input may be not satisfied by the limit solution (e.g. [9]).
Another feature of the technique is that to assign to each point a function of ε as weight, we need a one-to-one map between points and functions, which is usually achieved by indexing the points. This is both a drawback and an advantage: it is a drawback because the result depends of that indexing; it is an advantage because the indexing can be chosen freely to fit other purposes.

3 Contributions of this Paper

In this paper, we propose a simple method based on the algorithm using ears presented in § 2.1. Rather than using Devillers' algorithm [6], we choose a non-optimal variant that consists in maintaining dynamically a list of candidate ears and testing if their circumscribing ball is empty by applying the *in_sphere* predicate to *all the vertices* of the hole.
When an ear is found to belong to $\mathcal{DT}(\mathcal{S} \setminus \{v\})$, then the hole and the list of candidate ears are updated.

Devillers proves that if the Delaunay tetrahedra are added by increasing powers, then they are ears on the hole at the moment they are added. In our case, the tetrahedra are added in a different order. If we consider H_* the hole at some stage of the algorithm and T the Delaunay tetrahedron of smallest power inside H_*. If we consider H_T the hole before the addition of T in Devillers algorithm then by Devillers' result T is an ear of H_T, thus T is also an ear of H_* since $H_* \subset H_T$, which

proves that there is at least one Delaunay tetrahedron T which is an ear. So, in all cases, there will exist Delaunay ears in the list of candidate ears at each stage.

The number of candidates during the whole algorithm is $O(f)$, the test takes $O(k)$ for each ear, thus the whole complexity is $O(fk)$, where, as in § 2.1, k is the degree of the removed vertex and f is the number of created tetrahedra.

Since both f and k are small constants in general, this method will be quite efficient in practice. For example, the expected value of k when the input points follow a Poisson distribution is $\frac{96}{70}\pi^2+2 \simeq 15.5$ and the expected value of f is $\frac{72}{35}\pi^2 \simeq 20$ [15].
The reasons for choosing this sub-optimal algorithm are the following:

- The algorithms proposed by CGAL are parameterized by a *traits class* defining the basic geometric operations needed by the algorithms. The user can provides the algorithms with his/her own traits class. Therefore we need to keep the requirements for the traits class minimal. Devillers' algorithm would require an additional predicate for comparing the powers of a point with respect to two spheres, that is not necessary for the construction of the Delaunay triangulation.

- There are two ways of implementing Devillers' algorithm:
  - A predicate comparing the powers of a point with respect to two spheres defined by four vertices of the triangulation can be used. In this case, the same subexpressions are computed several times: the power of the removed point with the same sphere has to be compared with the power of the same point with several other spheres.
  - Avoiding redundant computations is possible by computing and storing the values of the powers. This leads to major problems: the fact of computing values instead of computing only signs yields important precision issues; an exact type could be used, which would give a bad efficiency. Filtering constructions such as in LOOK [12] could in principle be used to improve the running time, but combining such a mechanism while keeping the genericity of CGAL is still an open problem; this mechanism is not available yet in CGAL, which proposes only filtered predicates.

- In order to cope with degeneracies, Devillers' method would need a perturbation of the comparison of the power of a point with respect to different ears. These power tests have higher algebraic degree than the predicates that are really needed by the computation of the Delaunay triangulation, namely the *orient* and the *in_sphere* predicates (see § 5). A perturbation of these power tests would have to be compatible with the perturbation of the *in_sphere* predicate, which would be quite involved.

Thus, though this algorithm could be coded very efficiently as a stand-alone program, its integration in CGAL or in any library providing genericity and robustness is quite problematic. It seems to be better to use only the predicates that are intrinsic to the Delaunay triangulation, namely the *in_sphere* and the *orient* predicates, and to avoid any construction.

In the sequel, we focus on the perturbation used for solving degeneracies rather than on the algorithmic issues.

- The perturbation method presented in § 5 consists in adding a symbolic weight to each point. It does not create any flat tetrahedron. Allowing flat tetrahedra would have allowed us to use more trivial algorithms, but, as mentioned in § 4, we consider this as unacceptable both for theoretical and practical reasons.

- We also remark that the solution can be slightly modified in such a way that it does not depend on the insertion order of the points and defines the Delaunay triangulation uniquely even for degenerate configurations.

The method is implemented in CGAL. As far as we know, CGAL is the only publicly available software proposing a fully dynamic 3D Delaunay triangulation.

4 First Observations

4.1 Triangulation of a Simplicial Polyhedron. Let us consider a straight prism H with triangular basis such that its six vertices are cospherical. Assume that its rectangular facets are triangulated as shown in Figure 1. Let us now try to triangulate the interior of H. The six vertices of H are exactly in the same configuration regarding their incidences on H. Take one of them, say p *wlog*, then it can easily be seen that any possible tetrahedron having this vertex and any other three vertices of H will have an edge that crosses an existing edge on H.

4.2 Incremental Construction. Going back to the vertex removal in a 3D Delaunay triangulation, if the set of tetrahedra incident to a vertex v form the polyhedron H, then the previous remarks show that, when

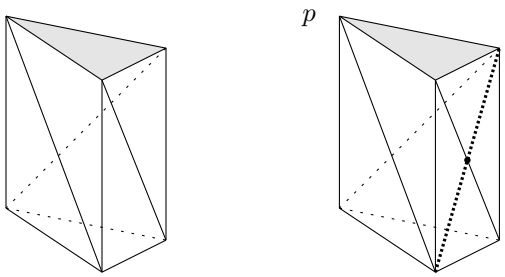

Figure 1: (1.1) has negative answer

v is removed, whatever choice will be made, retriangulating the hole H creates edges that cross edges of the tetrahedra outside the hole.
At first sight, the example of Figure 1 seems to prove that avoiding such edge crossings is impossible. In fact, this case cannot be built by an incremental algorithm coded as follows. As mentioned, the only problematic case is when there are cospherical (and coplanar at the same time) points. When a point p is inserted, the set of tetrahedra *conflicting* with p (defined as the tetrahedra whose circumscribing ball contains p in its interior) is determined, and these tetrahedra are deleted from the triangulation. If the algorithm considers as non-conflicting the tetrahedra whose circumscribing ball has p on its boundary, and thus if it does not delete these tetrahedra, then we get a unique construction of a Delaunay triangulation. This triangulation is unique for a given order of the points, but it depends on the order of insertion of the points.

Moreover, this incremental construction does not create any flat tetrahedron: if the point p to be inserted is coplanar with a triangle t that is a facet of the Delaunay triangulation, then the two tetrahedra incident to t will have the same conflict status with respect to p, which means that either they will both stay, and t will still be their common facet in the updated triangulation, or they will both be deleted and t will disappear. Thus, p will not form a flat tetrahedron together with t.

Assuming that the configuration shown in Figure 1 has been constructed by this incremental algorithm leads to the conclusion that the order of insertion of the points is cyclic, which is of course impossible. So, such a configuration cannot occur in our case.

Our problem can then be more precisely rephrased as: assuming that the Delaunay triangulation is constructed with the incremental algorithm considering each last inserted point p as not conflicting with tetrahedra whose vertices are cospherical with p, find the Delaunay triangulation of the hole that would have been computed if the removed point had never been inserted and if the other points had been inserted in the same order.

4.3 Allowing Flat Tetrahedra. Going back to the example in Figure 1, the new dashed edge that would be created to triangulate the prism would cross an edge of a tetrahedron that is exterior to the hole. The four vertices of these two crossing edges can be seen as a flat tetrahedron that could be added to the triangulation in order to make it combinatorially valid.
In the same way, if there are $k \geq 4$ cocircular vertices on some facet of the hole created when removing a vertex of the Delaunay triangulation, the Delaunay triangulation of the vertices of the hole is not uniquely defined, and retriangulating it may lead to two different triangulations of the same facet: one given by the interior tetrahedra, and one given by the exterior tetrahedra. It is well known that any triangulation in the plane can be obtained from any other triangulation having the same vertices by a sequence of $O(k^2)$ flips of diagonals of convex quadrilaterals [14]. Each such edge flip in 2D can be seen in 3D as a flat tetrahedron formed by the four *(coplanar)* vertices of the quadrilateral. Adding the $O(k^2)$ flat tetrahedra would yield a combinatorially valid triangulation.
Thus, allowing flat tetrahedra would allow us to get a simple solution to the problem. However, we do not consider this as an acceptable solution for the CGAL users. Indeed, these flat tetrahedra correspond neither with the definition of a Delaunay triangulation, since the circumscribing ball of a flat tetrahedron is not uniquely defined, nor to the usual intuition. Moreover, in practice, this would lead to a heavier user code: before applying geometric operations to a tetrahedron, such as computing its circumcenter, the user would have to check that the tetrahedron is not flat.

Since we know that a Delaunay triangulation without any flat tetrahedron always exists (it can be computed for instance using the incremental construction seen in § 4.2), our goal is to find a method that does not create any flat tetrahedron.

5 Perturbing the *in_sphere* Predicate

The *in_sphere* predicate is the basic predicate used when inserting or removing a point in a Delaunay triangulation.
Let p_0, p_1, p_2, p_3 be four non-coplanar points in $\mathbf{R}^3$.

in_sphere $(p_0, p_1, p_2, p_3, p_4)$

$$\left.\begin{array}{ll} > 0 & \text{if } p_4 \text{ is outside} \\ = 0 & \text{if } p_4 \text{ is on the boundary of} \\ < 0 & \text{if } p_4 \text{ is inside} \end{array}\right\} \text{ the ball circumscribing } p_0, p_1, p_2, p_3.$$

Let us notice straightaway that, if the triangulation admits no flat tetrahedron, then this predicate will be needed only when the points p_0, p_1, p_2, p_3 are actually non coplanar, and that no degenerate case has to be considered here: when inserting a new point p_4, it is only tested against tetrahedra that are already existing in the triangulation; this is also the case when filling the hole formed by the removal of another vertex, because we only consider ears of the hole formed by non coplanar facets, since the goal triangulation does not contain any flat tetrahedron.

The decision whether to create or delete a tetrahedron (p_0, p_1, p_2, p_3) in the Delaunay triangulation is easy when all the *in_sphere* tests against all points p_4 give > 0 and < 0 results. When 0 results appear, decisions must be made using other criteria.
Notice that coding the insertion as mentioned in § 4, by considering that the last inserted point is not conflicting with tetrahedra such that the result of the *in_sphere* test is 0 (i.e. considering that the point lies outside their circumscribing sphere), can be seen as using an implicit symbolic perturbation. It will be clear later that this perturbation corresponds to the perturbation proposed in the sequel.

It is well known that the *in_sphere* test can be computed in the following way:
in_sphere $(p_0, p_1, p_2, p_3, p_4) =$

$$\frac{sign\ Det(p_0, p_1, p_2, p_3, p_4)}{orient(p_0, p_1, p_2, p_3)}$$

where, if point p_i has coordinates (x_i, y_i, z_i) for each i,

$$orient(p_0, p_1, p_2, p_3) = sign \begin{vmatrix} 1 & 1 & 1 & 1 \\ x_0 & x_1 & x_2 & x_3 \\ y_0 & y_1 & y_2 & y_3 \\ z_0 & z_1 & z_2 & z_3 \end{vmatrix}$$

and $Det(p_0, p_1, p_2, p_3, p_4) =$

$$\begin{vmatrix} 1 & 1 & 1 & 1 & 1 \\ x_0 & x_1 & x_2 & x_3 & x_4 \\ y_0 & y_1 & y_2 & y_3 & y_4 \\ z_0 & z_1 & z_2 & z_3 & z_4 \\ x_0^2+y_0^2+z_0^2 & x_1^2+y_1^2+z_1^2 & x_2^2+y_2^2+z_2^2 & x_3^2+y_3^2+z_3^2 & x_4^2+y_4^2+z_4^2 \end{vmatrix}$$

The *sign* $Det(p_0, p_1, p_2, p_3, p_4)$ predicate in $\mathbf{R}^3$ can be seen as an orientation predicate in $\mathbf{R}^4$, if each point $p = (x, y, z)$ of $\mathbf{R}^3$ is projected onto a point $\pi(p) = (x, y, z, t)$ on the unit paraboloid Π of $\mathbf{R}^4$ with equation $t = x^2 + y^2 + z^2$ [11], [7].
Five points of $\mathbf{R}^3$ are cospherical if and only if their projections on Π lie in the same hyperplane of $\mathbf{R}^4$. The symbolic perturbation of the *in_sphere* test consists in adding respectively some value to the fourth coordinate of $\pi(p_0), \pi(p_1), \pi(p_2), \pi(p_3), \pi(p_4)$ so that these points are not in the same hyperplane any more in $\mathbf{R}^4$. Then the predicate answers positive or negative instead of zero.

Let $\mathcal{S}$ be the set of n points $\{p_0, p_1, \ldots, p_{n-1}\}$ of $\mathbf{R}^3$. We add here ε^{n-i} to the fourth coordinate of the point $\pi(p_i)$ of $\mathbf{R}^4$ corresponding to p_i. The quantity each point is perturbed with depends on its index. The way the points are indexed will be discussed in § 5.1.
$Det(p_i, p_j, p_k, p_l, p_m)$ is perturbed into
$Det_\varepsilon(p_i, p_j, p_k, p_l, p_m) =$

$$\begin{vmatrix} 1 & 1 & 1 & 1 & 1 \\ x_i & x_j & x_k & x_l & x_m \\ y_i & y_j & y_k & y_l & y_m \\ z_i & z_j & z_k & z_l & z_m \\ t_i+\varepsilon^{n-i} & t_j+\varepsilon^{n-j} & t_k+\varepsilon^{n-k} & t_l+\varepsilon^{n-l} & t_m+\varepsilon^{n-m} \end{vmatrix}$$

where

$$t_\star = x_\star^2 + y_\star^2 + z_\star^2 \text{ for } \star = i, j, k, l, m$$

Developing with respect to the last raw yields a polynomial in ε
$Det_\varepsilon(p_i, p_j, p_k, p_l, p_m) = Det(p_i, p_j, p_k, p_l, p_m) + orient(p_i, p_j, p_k, p_l)\varepsilon^{n-m} - orient(p_i, p_j, p_k, p_m)\varepsilon^{n-l} + orient(p_i, p_j, p_l, p_m)\varepsilon^{n-k} - orient(p_i, p_k, p_l, p_m)\varepsilon^{n-j} + orient(p_j, p_k, p_l, p_m)\varepsilon^{n-i}$
From now on, the five points p_i, p_j, p_k, p_l, p_m will be assumed to be non coplanar. As noticed at the beginning of this section, this assumption makes no restriction in practice since this is always the case when *in_sphere* is used.
When p_i, p_j, p_k, p_l, p_m are cospherical, the constant term $Det(p_i, p_j, p_k, p_l, p_m)$ of this polynomial vanishes to zero. At most one of the other coefficients *orient*() can be zero, otherwise two subsets with four points would consist of four coplanar points: either the five points would be coplanar, which contradicts our hypothesis, or the three common points of the two subsets would be collinear, which is impossible since these three points are cospherical. So, the polynomial Det_ε is not identically zero.

Let us examine the coefficients of Det_ε in order to determine the sign of $Det_\varepsilon(p_i, p_j, p_k, p_l, p_m)$.
If $m = \max\{i, j, k, l, m\}$

- if p_i, p_j, p_k, p_l are non coplanar,
  then $Det_\varepsilon(p_i, p_j, p_k, p_l, p_m)$ has the sign of
  $orient(p_i, p_j, p_k, p_l)$, thus
  in_sphere $(p_i, p_j, p_k, p_l, p_m) > 0$ and p_m lies out of the ball defined by p_i, p_j, p_k, p_l.

When five points are cospherical, the point with highest index among the five points will always be considered as lying outside the sphere defined by the four other points, if they are non coplanar.

- if p_i, p_j, p_k, p_l are coplanar,
  then, by hypothesis, p_m does not lie in the same plane. The predicate *in_sphere* $(p_i, p_j, p_k, p_l, p_m)$ has no geometric meaning since p_i, p_j, p_k, p_l do not define a sphere. Let us assume that $l = \max\{i, j, k, l\}$ (the other cases are similar, they just reduce to permutations in the columns of the determinant Det_ε). As noticed above, p_i, p_j, p_k, p_m define a sphere and *in_sphere* $(p_i, p_j, p_k, p_m, p_l)$ has a geometric meaning.

  $$Det_\varepsilon(p_i, p_j, p_k, p_m, p_l) = -Det_\varepsilon(p_i, p_j, p_k, p_l, p_m)$$

  has the sign of $orient(p_i, p_j, p_k, p_m)$.

  In fact, in the case when p_i, p_j, p_k, p_l are coplanar, the sign of *in_sphere* $(p_i, p_j, p_k, p_m, p_l)$ does not depend at all of the comparison between m and i, j, k, l. It depends only of the comparison between i, j, k, l. When four points are cocircular (coplanar and cospherical), the most perturbed point (the point with highest index) is considered to be out of the disk circumscribing the other three points in their common plane.

5.1 Indexing the Points. The current choice in CGAL for indexing the points is to use their order of insertion in the Delaunay triangulation. This choice corresponds to the implicit perturbation used for the insertion: at each step, the last point is more perturbed than all the previous ones, so, when it is cospherical with the four vertices of an already existing tetrahedron, it is considered as being outside the circumscribing ball.
Some users reported this choice as being annoying for their application [8]. In fact, any other way of indexing the points could be used. We are considering the possibility of changing the indexing in the future: we can choose for instance the lexicographical order on the points, or any other (such as the comparison of memory addresses). This change would be straightforward in the vertex removal since it is already coded using the perturbation scheme, but it would require a change in the insertion: the perturbation would have to be made explicit.
Choosing the lexicographical ordering of points would have the advantage of giving a unique definition of the Delaunay triangulation, even in degenerate cases. On the other hand, it may lead to a slower construction of the triangulation for some input. We may think of leaving the choice of the order to the user.

5.2 Weights and Regular Triangulation. As mentioned in § 2.2, the perturbation of *in_sphere* can also be seen as adding a weight ε^{n-i} to the points and computing the regular triangulation of the weighted points. This is an analogy with the "sliver exudation" method [4], [3] that consists in associating weights to points, chosen so that the *almost* flat tetrahedra that are unavoidable in a Delaunay triangulation disappear in the regular triangulation. In our case, the tetrahedra are not almost flat, but really flat, the weights are symbolic, and the triangulation we compute is a Delaunay triangulation without any flat tetrahedron.

6 Conclusion

We proposed a method for removing a vertex from a Delaunay triangulation, that works even in degenerate situations.
This method uses a symbolic perturbation technique that does not produce any flat tetrahedron. As can be seen in Appendix, the code for perturbing the *in_sphere* predicate is quite simple.
The solution is implemented in CGAL (releases 2.3 and 2.4).

Acknowledgments. The authors wish to thank Jeff Erickson for informal discussions at the very beginning of this research, Andreas Fabri for coding the ears algorithm of § 3 in the CGAL 3D triangulation package, Sylvain Pion for his improvements of the efficiency of this package, CGAL users, especially Tamal Dey and his students, for encouraging us in improving CGAL, and the members of the Prisme group at INRIA for preliminary discussions on the problem.

References

[1] A. Aggarwal, L. J. Guibas, J. Saxe, and P. W. Shor. A linear-time algorithm for computing the Voronoi diagram of a convex polygon. *Discrete Comput. Geom.*, 4(6):591–604, 1989.

[2] Pierre Alliez, Olivier Devillers, and Jack Snoeyink. Removing degeneracies by perturbing the problem or the world. *Reliable Computing*, 6:61–79, 2000. Special Issue on Computational Geometry, to appear.

[3] Siu-Wing Cheng and Tamal K. Dey. Quality meshing with weighted Delaunay refinement. In *Proc. 13th ACM-SIAM Sympos. Discrete Algorithms*, pages 137–146, 2002.

[4] Siu-Wing Cheng, Tamal K. Dey, Herbert Edelsbrunner, Michael A. Facello, and Shang-Hua Teng. Sliver exudation. In *Proc. 15th Annu. ACM Sympos. Comput. Geom.*, pages 1–13, 1999.

[5] L. P. Chew. Building Voronoi diagrams for convex polygons in linear expected time. Technical Report PCS-TR90-147, Dept. Math. Comput. Sci., Dartmouth College, Hanover, NH, 1986.

[6] Olivier Devillers. On deletion in Delaunay triangulation. *Internat. J. Comput. Geom. Appl.*, 12:193–205, 2002.

[7] Olivier Devillers, Stefan Meiser, and Monique Teillaud. The space of spheres, a geometric tool to unify duality results on Voronoi diagrams. In *Proc. 4th Canad. Conf. Comput. Geom.*, pages 263–268, 1992. Technical Report INRIA 1620, http://www.inria.fr/rrrt/rr-1620.html.

[8] Tamal K. Dey and Joachim Giesen. Detecting undersampling in surface reconstruction. In *Proc. 17th Annu. ACM Sympos. Comput. Geom.*, pages 257–263, 2001.

[9] M. B. Dillencourt. Realizability of Delaunay triangulations. *Inform. Process. Lett.*, 33(6):283–287, February 1990.

[10] H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graph.*, 9(1):66–104, 1990.

[11] H. Edelsbrunner and R. Seidel. Voronoi diagrams and arrangements. *Discrete Comput. Geom.*, 1:25–44, 1986.

[12] Stefan Funke and Kurt Mehlhorn. Look: A lazy object-oriented kernel for geometric computation. In *Proc. 16th Annu. ACM Sympos. Comput. Geom.*, pages 156–165, 2000.

[13] M. Heller. Triangulation algorithms for adaptive terrain modeling. In *Proc. 4th Internat. Sympos. Spatial Data Handling*, pages 163–174, 1990.

[14] F. Hurtado, M. Noy, and J. Urrutia. Flipping edges in triangulations. *Discrete Comput. Geom.*, 22(3):333–346, 1999.

[15] Atsuyuki Okabe, Barry Boots, and Kokichi Sugihara. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, Chichester, UK, 1992.

[16] R. Seidel. The nature and meaning of perturbations in geometric computing. *Discrete Comput. Geom.*, 19:1–17, 1998.

Appendix - CGAL Code for the Perturbed Predicate

```
template < class Gt, class Tds >
Bounded_side
Delaunay_triangulation_3<Gt,Tds>::
side_of_sphere_finite_perturb
            (Vertex_handle v0, Vertex_handle v1,
             Vertex_handle v2, Vertex_handle v3,
             Vertex_handle v) const
  // v0,v1,v2 and v0,v1,v3 are supposed to form two
  // finite facets of the triangulation, forming a
  // convex angle must not be used otherwise
{
  CGAL_triangulation_precondition
       ( (dimension() == 3) &&
         (! is_infinite(v0)) && (! is_infinite(v1)) &&
         (! is_infinite(v2)) && (! is_infinite(v3)) );

  const Point & p0 = v0->point();
  const Point & p1 = v1->point();
  const Point & p2 = v2->point();
  const Point & p3 = v3->point();

  CGAL_triangulation_precondition
              ( orientation(p0,p1,p2,p3) == POSITIVE );

  if (is_infinite(v))
      return ON_UNBOUNDED_SIDE;

  const Point & p = v->point();

  Bounded_side bs =
   Bounded_side(side_of_oriented_sphere(p0,p1,p2,p3,p));
  // side_of_oriented_sphere is the non perturbed
  // in_sphere predicate

  if ( bs != ON_BOUNDARY )
      return bs;

  int i0 = v0->get_order_of_creation();
  int i1 = v1->get_order_of_creation();
  int i2 = v2->get_order_of_creation();
  int i3 = v3->get_order_of_creation();
  int i = v->get_order_of_creation();
  Orientation o;

  int m =
  std::max(i0,std::max(i1,std::max(i2,std::max(i3,i))));
  // we look whether the leading monomial of the
  // determinant has non null coefficient
  if (m == i)
    return ON_UNBOUNDED_SIDE;
  // since p0 p1 p2 p3 are supposed to be non coplanar
  // and positively oriented
  if (m == i3 && (o = orientation(p0,p1,p2,p)) != ZERO )
    return Bounded_side(o);
  if (m == i2 && (o = orientation(p0,p1,p3,p)) != ZERO )
    return Bounded_side(-o);
  if (m == i1 && (o = orientation(p0,p2,p3,p)) != ZERO )
    return Bounded_side(o);
  if (m == i0 && (o = orientation(p1,p2,p3,p)) != ZERO )
    return Bounded_side(-o);

  // if not yet returned, then the leading monomial
  // of the determinant has null coefficient
  // we look whether the 2nd monomial has
  // non null coefficient
  m = maxless(i0,i1,i2,i3,i,m);
  if (m == i)
    return ON_UNBOUNDED_SIDE;
  if (m == i3 && (o = orientation(p0,p1,p2,p)) != ZERO )
    return Bounded_side(o);
  if (m == i2 && (o = orientation(p0,p1,p3,p)) != ZERO )
    return Bounded_side(-o);
  if (m == i1 && (o = orientation(p0,p2,p3,p)) != ZERO )
    return Bounded_side(o);
  if (m == i0 && (o = orientation(p1,p2,p3,p)) != ZERO )
    return Bounded_side(-o);

  // as seen in Section 5, at most one orientation test
  // is zero, so a result has been returned
}
```