Flipping to Robustly Delete a Vertex in a Delaunay Tetrahedralization

Hugo Ledoux[1], Christopher M. Gold[1], and George Baciu[2]

[1] GIS Research Centre, School of Computing,
University of Glamorgan, Pontypridd,
CF37 1DL, Wales, UK
hledoux@glam.ac.uk, christophergold@voronoi.com
[2] Department of Computing,
Hong Kong Polytechnic University, Hong Kong
csgeorge@comp.polyu.edu.hk

Abstract. We discuss the deletion of a single vertex in a Delaunay tetrahedralization (DT). While some theoretical solutions exist for this problem, the many degeneracies in three dimensions make them impossible to be implemented without the use of extra mechanisms. In this paper, we present an algorithm that uses a sequence of bistellar flips to delete a vertex in a DT, and we present two different mechanisms to ensure its robustness.

1 Introduction

The construction of the Delaunay tetrahedralization (DT) of a set S of points in the Euclidean space $\mathbb{R}^3$ is a well-known problem and many efficient algorithms exist [1, 2, 3]. The 'inverse' problem — the deletion of a vertex v in a DT(S), thus obtaining DT($S \setminus \{v\}$) — is however much less documented and is still a problem in practice. Most of the work on this topic has been done for the two-dimensional case and very little can be found for the three- and higher-dimensional cases. The problem has been tackled mostly by removing from the triangulation all the simplices incident to v and retriangulating the 'hole' thus formed (see Fig. 1 for the 2D case). Throughout this paper, we denote by star(v) the star-shaped polytope formed by the union of all the simplices incident to a vertex v in a d-dimensional Delaunay triangulation.

An optimal solution exists for the 2D problem [4], but sub-optimal algorithms are nevertheless usually preferred for an implementation because of their simplicity and because the average degree k of a vertex in a 2D Delaunay triangulation is only 6. The most elegant of these algorithms is due to Devillers [5], who transforms the problem into the construction of the convex hull of the points on the boundary of star(v) lifted onto the paraboloid in 3D. Mostafavi et al. [6] propose a simpler algorithm, where each triangle used for the retriangulation is tested against each vertex of star(v). These two algorithms have respectively a time complexity of O(k log k) and O(k^2).

O. Gervasi et al. (Eds.): ICCSA 2005, LNCS 3480, pp. 737–747, 2005.

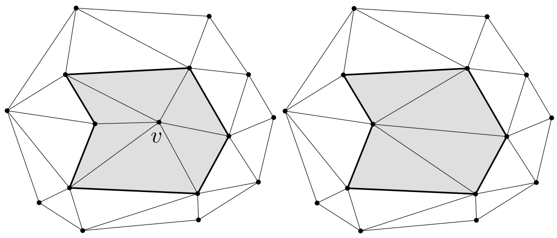


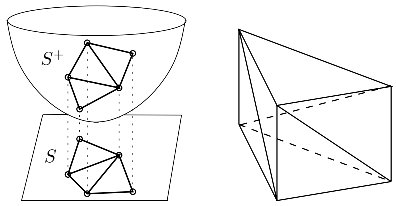
Fig. 1. Delaunay triangulations before and after vertex v has been deleted

Most algorithms in computational geometry assume that inputs are in 'general position', and the handling of degeneracies are usually left to the programmers. Modifying the original algorithm to make it robust for any input can be in some cases an intricate and error-prone task. Luckily, the deletion of a single vertex in a 2D DT does not have many special cases and its robust implementation is quite simple. However, the numerous degeneracies make the implementation of Devillers' and Mostafavi et al.'s algorithms impossible in 3D (impossible without the use of an extra mechanism that is), despite the facts that the former proved that his algorithm is valid in any dimensions, and that common sense suggests the latter algorithm generalises easily. The problems are caused by the fact that not every polyhedron can be tetrahedralized, as explained in Sect. 2. Devillers and Teillaud [7] recognised that and used perturbations [8] to solve the problem. Unfortunately, the major part of their paper is devoted to explaining the perturbation scheme and few details about the algorithm are given.

In this paper, we describe an algorithm to delete a vertex in a DT, and we show in Sect. 4 that, instead of creating a 'hole' in the tetrahedralization, it is possible to tackle the problem differently and use bistellar flips; the flips needed are described in Sect. 3. Flipping permits us to keep a complete tetrahedralization during the whole deletion process, and hence the algorithm is relatively simple to implement and numerically more robust. We also discuss in Sect. 5 two different methods to ensure the algorithm is robust against degenerate cases. The first one uses a symbolic perturbation scheme, and the second is an empirical method that requires the modification of some tetrahedra outside $star(v)$.

2 Delaunay Tetrahedralization

A Delaunay tetrahedralization (DT) of a set S of points in $\mathbb{R}^3$ is a set of non-overlapping tetrahedra, which have *empty* circumspheres, whose union completely fills the convex hull ($\mathcal{CH}$) of S. It can be constructed incrementally [3], with an algorithm based on the divide-and-conquer paradigm [2], or even by transforming the problem into the construction of a four-dimensional convex hull [9]. Another method consists of using bistellar flips (see Sect. 3) to modify the configuration of adjacent tetrahedra after the insertion of a new point. The major problem when designing flip-based algorithms in 3D is that even if some



(a) (b)

Fig. 2. **(a)** The parabolic lifting map of a set S of points in the plane. Each triangular face of the convex hull of the lifted set S^+ in 3D corresponds to a triangle of the 2D Delaunay triangulation of S. **(b)** The Schönhardt polyhedron is impossible to tetrahedralize

adjacent tetrahedra need to be modified, it is not always possible to flip them, as Joe [10] proves. He nevertheless later proved that if a single point v is added to a valid DT(S), then there always exists at least one sequence of flips to construct DT($S \cup \{v\}$) [1]. These results have also been generalised to $\mathbb{R}^d$ [11].

Constructing DT(S) essentially requires two geometric predicates: *Orient*, which determines on what side of a plane a point lies; and *InSphere*, which determines if a point p is inside, outside or lies on a sphere. The *InSphere* test is derived from the well-known *parabolic lifting map* [9], which describes the relationship that exists between a d-dimensional Delaunay triangulation and a convex hull in $(d+1)$ dimensions (see Fig. 2(a)).

While any polygon in 2D can be triangulated, some arbitrary polyhedra, even if they are star-shaped, cannot be tetrahedralized without the addition of extra vertices, the so-called Steiner points. Fig. 2(b) shows an example, as it was first illustrated by Schönhardt [12]. In this paper, we are interested in a special case of the tetrahedralization problem: the polyhedron is star(v), a star-shaped polyhedron (not necessarily convex), formed by all the tetrahedra in DT(S) incident to the vertex v. The tetrahedralization of star(v) is always possible, and moreover with locally Delaunay tetrahedra. Let $\mathcal{T}$ be a DT(S). If v is added to $\mathcal{T}$, thus getting $\mathcal{T}^v = \mathcal{T} \cup \{v\}$, with an incremental insertion algorithm [1, 3], all the tetrahedra in $\mathcal{T}$ whose circumspheres contain v will be deleted; the union of these tetrahedra forms a polyhedron P. Then, P will be retetrahedralized with many tetrahedra all incident to v. Now consider the deletion of v from $\mathcal{T}^v$. Notice that v could actually be any vertices in S, as a DT is unique and not affected by the order of insertion. The polyhedron star(v) is exactly the same as P, therefore $\mathcal{T}$ tetrahedralize P.

3 Three-Dimensional Bistellar Flips

A bistellar flip is a local topological operation that modifies the configuration of some adjacent tetrahedra. As shown in Lawson [13], there exist four different

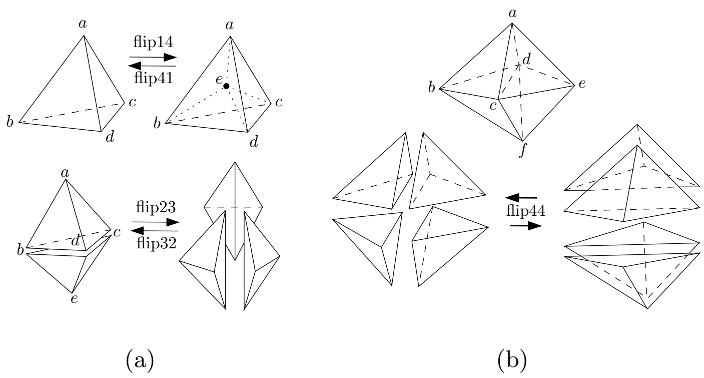


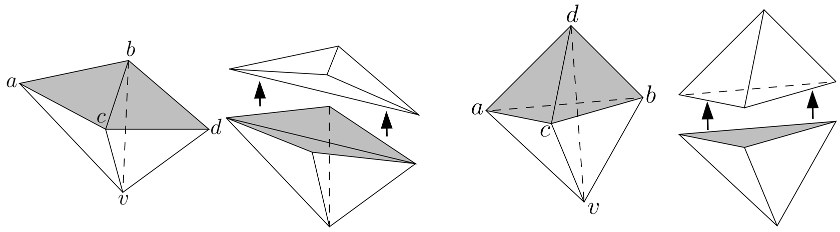
Fig. 3. (a) Three-dimensional bistellar flips. **(b)** Degenerate *flip44*

flips in $\mathbb{R}^3$, based on the different configurations of a set $S = \{a, b, c, d, e\}$ of points in general position: *flip14*, *flip41*, *flip23* and *flip32* (the numbers refer to the number of tetrahedra before and after the flip). These flips are illustrated in Fig. 3(a). When the five points of S lie on the boundary of $\mathcal{CH}(S)$, S can be tetrahedralized with two or three tetrahedra, and the *flip23* and *flip32* are the operations that substitute one tetrahedralization by another one. A *flip14* refers to the operation of inserting a vertex inside a tetrahedron, and splitting it into four tetrahedra; and a *flip41* is the inverse operation that deletes a vertex.

To deal with degenerate cases, other flips need to be defined. Shewchuk [14] defines, and uses for the construction of constrained Delaunay triangulations, degenerate flips. A flip is said to be degenerate if it is a non-degenerate flip in a lower dimension. It is used for special cases such as when a new point is inserted directly onto an edge or a face of a triangular face. Consider the set $S = \{a, b, c, d, e, f\}$ of points configured as shown in Fig. 3(b), with points b, c, d and e being coplanar. If S is tetrahedralized with four tetrahedra all incident to one edge — this configuration is called the *config44* — then a flip44 transforms one tetrahedralization into another one also having four tetrahedra. Note that the four tetrahedra are in *config44* before and after the flip44. A flip44 is actually a combination in one step of a flip23 (that creates a flat tetrahedron) followed immediately by a flip32 that deletes the flat tetrahedron; a flat tetrahedron is a tetrahedron spanned by four coplanar vertices (its volume is zero). We show in Sect. 5 why it is necessary when deleting a vertex.

4 Flipping to Delete a Vertex

This section describes an algorithm for deleting a vertex v in a Delaunay tetrahedralization $\mathrm{DT}(S)$ of a set S of points in general position. Unlike other known approaches, the polyhedron $\mathrm{star}(v)$ is not deleted from $\mathrm{DT}(S)$; $\mathrm{DT}(S \setminus \{v\})$ is obtained by restructuring $\mathrm{star}(v)$ with a sequence of bistellar flips.



(a) (b)

Fig. 4. Flipping of an ear. In both cases, link(v), before and after the flip, is represented by the shaded triangular faces. **(a)** A 2-ear *abcd* is flipped with a flip23. **(b)** A 3-ear *abcd* is flipped by a flip32

4.1 Ears of a Polyhedron

Let P be a polyhedron that is made up of triangular faces. An ear of P is a potential, or 'imaginary' tetrahedron, that could be used to tetrahedralize P; this is the three-dimensional equivalent of the ear of a polygon as used in deletion algorithms (see e.g. [5]). Referring to the Fig. 4, we can affirm that there exist two kinds of ears for P: a *2-ear* is formed by two adjacent triangular faces *abc* and *bcd* sharing edge *bc*; and a *3-ear* is formed by three adjacent triangular faces *abd*, *acd* and *bcd* sharing vertex d. A 3-ear is actually formed by three 2-ears overlapping each other. On the surface of P, every 2-ear has four neighbouring ears and every 3-ear has three; in both cases, these neighbours can either be 2- or 3-ears. Not every pair of adjacent faces of P is considered as a *valid* ear. A 2-ear is valid if and only if the line segment *ad* is inside P; and a 3-ear is valid if and only if the triangular face *abc* is inside P. In the case of the deletion of a vertex v in a DT, P is a star-shaped polyhedron star(v). An ear of star(v) is valid if it is convex outwards from v; this can be tested with two *Orient* tests.

4.2 Flipping an Ear

Let link(v) be the union of the triangular faces on the boundary of star(v) which are not incident to vertex v. Consider an ear ε (formed by two or three adjacent triangular faces in link(v)) of the polyhedron star(v). Flipping ε means creating in the tetrahedralization the tetrahedron spanned by the four vertices of ε, by flipping the two or three tetrahedra that define ε. As shown in Fig. 4, different flips are applied to different types of ears:

1. if ε is a 2-ear *abcd*, defined by the tetrahedra *abcv* and *bcdv*, then a flip23 creates the tetrahedra *abcd*, *abvd* and *acvd*. The results of that flip23 are that, first, the tetrahedron *abcd* is not part of star(v) anymore; and second, link(v) is modified as ε is replaced by an ear ε' formed by the triangular faces *abd* and *acd*. A 2-ear is said to be *flippable* if and only if the union of the two tetrahedra defining it is a convex polyhedron.

2. if ε is a 3-ear $abcd$, defined by the tetrahedra $abdv$, $bcdv$ and $acdv$, then a flip32 creates the tetrahedra $abcd$ and $abcv$. After the flip, the tetrahedron $abcd$ is not part of star(v) anymore, and link(v) has the face abc instead of the three faces before the flip32. Also, the degree of v is reduced by 1 by such a flip. A 3-ear is *flippable* if and only if vertices d and v are on each side of the face abc, i.e. that v should the 'outside' ε.

4.3 Deletion with *InSphere*

The algorithm we describe in this section, called DeleteInSphere, is a generalisation of Mostafavi et al.'s [6] in 3D, and proceeds as follows. First, all the ears of star(v) are built and stored in a simple dynamic list. The idea of the algorithm is to take an ear ε from the list (any ear) and process it if these three conditions are respected: ε is valid, flippable and locally Delaunay. An ear ε is locally Delaunay if its circumsphere does not contain any other vertices on the boundary of star(v); this is tested with *InSphere* tests. There is no particular order in which the ears are processed. If an ear respects the three conditions, it is flipped, and, as a result, star(v) is modified and a tetrahedron spanned by the four vertices of ε is added to $\mathrm{DT}(S \setminus \{v\})$. As flips are performed, star(v) 'shrinks' and the configuration of the tetrahedra inside it changes; thus non-valid ears become valid, and vice versa. If an ear does not respect one of the three conditions, the next ear in the list is tested. The time complexity of the algorithm is $\mathrm{O}(t\,k)$: t tetrahedra are created to retetrahedralize star(v), and each of these tetrahedra must be tested against the k vertices on the boundary of star(v). A flip is assumed to be performed in constant time because only a finite number of adjacent tetrahedra are involved.

The correctness proof of the algorithm is omitted here, but here is the main idea. Let $\mathcal{T}$ be a DT, and let $\mathcal{T}^v = \mathcal{T} \cup \{v\}$ be the tetrahedralization after the insertion of v with a flip-based algorithm. Joe [1] proves that v can always be inserted in $\mathcal{T}$ by a sequence of bistellar flips; each flip will remove from $\mathcal{T}$ exactly one tetrahedron whose circumsphere contains v. Although some locally non-Delaunay tetrahedra will be impossible to flip, there will always be at least one possible flip at each step of the algorithm. Consider now the deletion of v in $\mathcal{T}^v$, thus getting $\mathcal{T}' = \mathcal{T}^v \setminus \{v\}$. Sect. 3 shows that each flip has its inverse, therefore reversing the flips used to construct $\mathcal{T}^v$ trivially constructs $\mathcal{T}'$. Also, notice that $\mathcal{T}'$ must be equal to $\mathcal{T}$ since we assume general position. This means that, to construct $\mathcal{T}'$, only the inverse flips of the flips used to construct $\mathcal{T}^v$ can be used. Thus, constructing $\mathcal{T}'$ can be seen as exactly the inverse of constructing $\mathcal{T}^v$, and, as a result, at each step of the process a locally Delaunay ear will be flippable.

5 Degeneracies

Degenerate cases occur when one of the following two conditions arises: the vertex v to be deleted lies on the boundary of $\mathcal{CH}(S)$; the set of points S is not in general position, that is four or more points are coplanar, and/or five or more points are cospherical.

The algorithm DeleteInSphere as described in Sect. 4 is not valid for deleting a vertex on the boundary of $\mathcal{CH}(S)$. This case can nevertheless be easily avoided by starting the construction algorithm of a DT with four non-coplanar points forming a tetrahedron big enough to contain all the points in S.

The coplanarity of points in S leads to the use of the degenerate bistellar *flip44*; we explain in Sect. 5.1 below how it is used.

When five or more points in S are cospherical, DT(S) is not unique. Let $\mathcal{T}$ be a DT(S) containing five or more cospherical points, and let v be a point located 'inside' the cospherical points. Consider the insertion of v in $\mathcal{T}$, thus getting $\mathcal{T}^v$, followed immediately by its deletion to get the tetrahedralization $\mathcal{T}'$. The tetrahedralization $\mathcal{T}'$, although being a valid DT, will not necessarily be the same as $\mathcal{T}$. Cospherical points will introduce an ambiguity as to which flips should be performed to delete v in $\mathcal{T}^v$. A flip used to delete v from $\mathcal{T}^v$, although possible and performed on a Delaunay ear, is not necessarily the inverse of a flip that was used to construct $\mathcal{T}^v$. Unfortunately, one or more of these 'wrong' flips can lead to a polyhedron star(v) that is impossible to tetrahedralize.

There exist two solutions to deal with this problem. The first one is to *prevent* an untetrahedralizable polyhedron by perturbing vertices to ensure that a DT is unique even for degenerate inputs. This is briefly described in Sect. 5.2. The main disadvantage of this method is that the same perturbation scheme must be used for all the operations performed on a DT. As a result, if one only has a DT and does not know what perturbation scheme was used to create it, then this method cannot be used. It is of course always possible to modify a DT so that it is consistent with a given perturbation scheme, but that could require a lot of work in some cases. Also, for some applications, using perturbations is not always possible. An example is a modelling system where points are moving while the topological relationships in the DT are maintained. It would be quite involved to ensure that the DT is consistent at all times with a perturbation scheme. The alternative solution consists of *recovering* from an untetrahedralizable star(v) by modifying the configuration of some triangular faces on its boundary. Such an operation requires the modification of some tetrahedra outside star(v). The implementation of this method is greatly simplified if a complete tetrahedralization is kept because the 'outside' tetrahedra can simply be flipped. The method is called 'unflipping' and is described in Sect. 5.3.

5.1 Handling Coplanar Points with the Degenerate *flip44*

Let ε be a 2-ear of a polyhedron star(v), and let τ_1 and τ_2 be the two tetrahedra defining ε. Consider τ_1 and τ_2 to have four coplanar vertices and ε to be Delaunay (e.g. in Fig. 4(a), *abdv* would be coplanar). A simple flip23 on ε is not always possible since it creates a flat tetrahedron; actually the flip23 is possible if and only if τ_1 and τ_2 are in *config44* with two tetrahedra τ_3 and τ_4 that are inside star(v) and define an ear ε_2 that is Delaunay. In that case, a flip44 will flip in one step both ε and ε_2. If τ_1 and τ_2 are not in *config44*, then ε cannot be flipped.

If four or more coplanar vertices are present in a DT, one must be aware of the presence of flat tetrahedra. They can be created during the process of updating a DT (after a flip), but no flat tetrahedron can exist in a DT (it violates the Delaunay criterion since the circumsphere of a flat tetrahedron is undefined). It is known that a DT without any flat tetrahedra always exists. In DeleteInSphere, flat tetrahedra are permitted only if they are incident to v; an ear clearly cannot be flat because after being processed it becomes a tetrahedron of DT.

5.2 Symbolic Perturbations to Handle Cospherical Points

Perturbing a set S of points means moving the points by an infinitesimal amount to ensure that S is in general position. Unfortunately, moving points in $\mathbb{R}^d$ can have serious drawbacks: tetrahedra that are valid in the perturbed set of points can become degenerate (e.g. flat) when the points are put back to their original position. In the case of the deletion of a vertex v in a DT in $\mathbb{R}^3$, only cospherical points really cause problems — they can lead to an untetrahedralizable polyhedron — since coplanarity can be handled with the flip44. A method to perturb only cospherical points without actually moving them was proposed in Edelsbrunner and Mücke [15–Sect. 5.4]. It involves perturbing the points in $\mathbb{R}^{d+1}$ by using the parabolic lifting map. In $\mathbb{R}^3$, five points are cospherical if and only if the five lifted points are coplanar on the paraboloid in $\mathbb{R}^4$. Thus, each point of S is perturbed by a (very) small amount so that no five points in $\mathbb{R}^4$ lie on the same hyperplane. The method cannot be applied just for the deletion of a single vertex since the resulting tetrahedralization of star(v) would not necessarily be consistent with the tetrahedralization outside star(v). The main goal of using this method is having a unique DT even when five or more points in S are cospherical, so that there is a clear ordering of the flips to perform to delete v. The same perturbation scheme must therefore be used for every operation performed on the DT, including its construction.

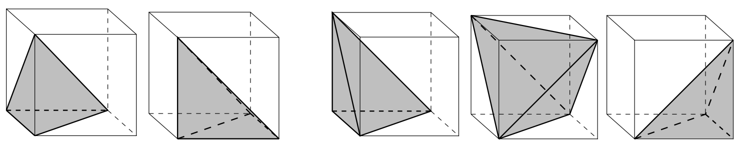
For a very easy implementation of this perturbation scheme, see [16]. With this method, the amount by which each point is moved does not have to be calculated explicitly because the perturbations are implemented symbolically in the *InSphere* test. It should be noted that the `CGAL` library uses the same scheme [7], although the implementation is different.

5.3 Unflipping

We have tested the algorithm DeleteInSphere with many different datasets — points randomly distributed in a cube or a sphere, lying on the boundary of regular solids (spheres, cubes, paraboloid, etc.), and also geologic datasets with boreholes — and, in our experience, the case of an untetrahedralizable polyhedron occurs only when most of the points in S are both coplanar and cospherical, that is when the spacing between points in the $x-y-z$ directions is constant, as in a 3D grid. Unflipping means modifying some triangular faces on the boundary of star(v) so that a tetrahedralization of star(v) is possible. For example, the most 'common' untetrahedralizable star(v) during our tests was very similar to the Schönhardt polyhedron depicted in Fig. 2(b), except that the three quadrilateral faces are 'flat': they are formed by two coplanar triangular faces. Notice that if only one diagonal of a quadrilateral face is flipped (think of a *flip22* in a 2D triangulation) then the polyhedron can easily be tetrahedralized with three tetrahedra. Thus, to recover from an untetrahedralizable star(v), we propose flipping the diagonal of one flat 2-ear of star(v) and continue the deletion process as usual afterwards. This is an iterative solution: one flip might be sufficient in some cases, but if another untetrahedralizable star(v) is later obtained then another diagonal must be flipped. Flipping the diagonal of a flat ear obviously involves the modification of the tetrahedra, inside and outside star(v), incident to the diagonal edge. The only way to do this is with a flip44 involving the two tetrahedra τ_1 and τ_2 forming the flat ear and two tetrahedra τ_1' and τ_2' adjacent to them outside star(v). The assumption behind this method is the following. An untetrahedralizable star(v) occurs only when most of the points in S form a 3D grid (S can also be seen as being formed by many adjacent cubes formed by eight vertices), and therefore one of the flat ears of star(v) will be incident to one such cube. The modification of the tetrahedralization of a cube is allowed since its eight vertices are cospherical.

The configuration of tetrahedra outside star(v) and incident to a flat ear ε will not always be the same, as there are many ways to tetrahedralize a cube. The two most common configurations are as follows. In the first configuration, only two tetrahedra τ_1' and τ_2' are adjacent to ε (see Fig. 5(a)), thus forming a *config44*. A flip44 is then allowed if the five vertices of τ_1' and τ_2' are cospherical. In the second configuration, three tetrahedra are incident to ε, as in Fig. 5(b). Then if the six vertices of the three tetrahedra are cospherical, a flip23 on two of the three tetrahedra will modify the configuration such that only two tetrahedra are incident to ε. A flip44 is then possible.

Other configurations can occur and by flipping locally it is possible to obtain two tetrahedra incident to a flat ear. In our experience, when star(v) is an untetrahedralizable polyhedron, there is always a flat ear whose diagonal can be flipped such that DeleteInSphere makes progress towards the deletion of v.



(a) (b)

Fig. 5. Two possible configurations when unflipping. The flat ear ε is the bottom face of the cube. **(a)** Two tetrahedra are incident to ε. **(b)** Three tetrahedra are incident to ε

6 Discussion

In brief, to make DeleteInSphere robust for any configuration of data, several things must be done. First, the coplanarity of vertices must be handled with the degenerate *flip44*. Second, cospherical vertices, which may lead to an untetrahedralizable polyhedron, can be handled with either symbolic perturbations or with the unflipping method. It should be noticed that the implementation of a perturbation scheme is effective only if exact arithmetic is used for all the predicates involved.

Acknowledgements. Part of this work was done when the first author was at the Hong Kong Polytechnic University, and we thank the support of the Hong Kong's Research Grants Council (Project PolyU 5068/00E).

References

1. Joe, B.: Construction of three-dimensional Delaunay triangulations using local transformations. Computer Aided Geometric Design **8** (1991) 123–142
2. Cignoni, P., Montani, C., Scopigno, R.: DeWall: a fast divide & conquer Delaunay triangulation algorithm in E^d. Computer-Aided Design **30** (1998) 333–341
3. Watson, D.F.: Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. Computer Journal **24** (1981) 167–172
4. Chew, L.P.: Building Voronoi diagrams for convex polygons in linear expected time. Technical Report PCS-TR90-147, Dartmouth College, Computer Science, Hanover, NH (1990)
5. Devillers, O.: On deletion in Delaunay triangulations. International Journal of Computational Geometry and Applications **12** (2002) 193–205
6. Mostafavi, M.A., Gold, C.M., Dakowicz, M.: Delete and insert operations in Voronoi/Delaunay methods and applications. Computers & Geosciences **29** (2003) 523–530
7. Devillers, O., Teillaud, M.: Perturbations and vertex removal in a 3D Delaunay triangulation. In: Proceedings 14th ACM-SIAM Symposium on Discrete Algorithms (SODA), Baltimore, MD, USA (2003) 313–319
8. Seidel, R.: The nature and meaning of perturbations in geometric computing. Discrete & Computational Geometry **19** (1998) 1–17
9. Edelsbrunner, H., Seidel, R.: Voronoi diagrams and arrangements. Discrete & Computational Geometry **1** (1986) 25–44
10. Joe, B.: Three-dimensional triangulations from local transformations. SIAM Journal on Scientific and Statistical Computing **10** (1989) 718–741
11. Rajan, V.T.: Optimality of the Delaunay triangulation in $\mathbb{R}^d$. In: Proceedings 7th Annual Symposium on Computational Geometry, North Conway, New Hampshire, USA, ACM Press (1991) 357–363
12. Schönhardt, E.: Über die zerlegung von dreieckspolyedern in tetraeder. Mathematische Annalen **98** (1928) 309–312
13. Lawson, C.L.: Properties of n-dimensional triangulations. Computer Aided Geometric Design **3** (1986) 231–246

14. Shewchuk, J.R.: Updating and constructing constrained Delaunay and constrained regular triangulations by flips. In: Proceedings 19th Annual Symposium on Computational Geometry, San Diego, USA, ACM Press (2003) 181–190
15. Edelsbrunner, H., Mücke, E.P.: Simulation of Simplicity: a technique to cope with degenerate cases in geometric algorithms. ACM Transactions on Graphics **9** (1990) 66–104
16. Shewchuk, J.R.: Sweep algorithms for constructing higher-dimensional constrained Delaunay triangulations. In: Proceedings 16th Annual Symposium on Computational Geometry, Hong Kong, ACM Press (2000) 350–359