Regular Triangulations of Dynamic Sets of Points

Marc Vigo, Núria Pla
marc@lsi.upc.es, nuria@lsi.upc.es
Dept. de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Barcelona

20 October 2000

Abstract

The Delaunay triangulations of a set of points are a class of triangulations which play an important role in a variety of different disciplines of science. Regular triangulations are a generalization of Delaunay triangulations that maintain both their relationship with convex hulls and with Voronoi diagrams. In regular triangulations, a real value, its weight, is assigned to each point.

In this paper a simple data structure is presented that allows regular triangulations of sets of points to be dynamically updated, that is, new points can be incrementally inserted in the set and old points can be deleted from it. The algorithms we propose for insertion and deletion are based on a geometrical interpretation of the history data structure in one more dimension and use lifted flips as the unique topological operation. This results in rather simple and efficient algorithms. The algorithms have been implemented and experimental results are given.

Keywords: Delaunay and regular triangulations, Voronoi diagrams, convex hulls.

1 Introduction

Triangulations play a fundamental role in Computer Graphics, and the computation of the Delaunay triangulation of a set of points and its dual, the Voronoi diagram, is one of the classical problems of Computational Geometry. Their properties, as well as algorithms for constructing them, are extensively covered in textbooks, such as [17], [7] or [16], and also in survey papers [1] or [11]. Regular triangulations are a well known extension of Delaunay triangulations of a set of points in $\mathbb{R}^d$, which is obtained assigning a scalar value, its weight, to each point. The regular triangulation of a set of points reduces to Delaunay triangulation of the set when all the weights have the same value. Regular triangulations bridge geometric and algebraic issues, for example they have a close relationship with Gröbner bases and also with the theory of discriminants, hypergeometric functions, etc [14]. Furthermore, there exists a close connection between regular triangulations in $\mathbb{R}^d$ and convex hulls in $\mathbb{R}^{d+1}$, based on the paraboloid mapping of the points $\mathbb{R}^d$ in one more dimension. The algorithms we present in this paper make an intensive use of this map.

A number of incremental algorithms have been proposed for obtaining Delaunay triangulations, based on local topological transformations, called flips (or swaps); historically, the first author to use this approach, for a set of points in the plane, was Lawson [13], but many improvements and generalizations have been proposed later, see for example [12] or [2]. Edelsbrunner and Shah extended the flipping algorithm to work for regular triangulations in d dimensions, [8, 10]. These algorithms are designed for incrementally including new points in a regular (or Delaunay) triangulation, but only a few allow deleting points (dynamically updating the related data structures). Devillers [6] enumerates several algorithms for deleting vertices from a Delaunay triangulation, and proposes a simple way for removing a single vertex from a two-dimensional Delaunay triangulation. His method consists on retriangulating the interior of the star-shaped polygon around vertex to be deleted, which is equivalent to swapping edges around the vertex in a particular order. The algorithm can be generalized quite easily to work in d dimensions.

In this paper we propose a simple data structure that allows regular triangulations of sets of points to be dynamically updated, that is, new points can be inserted and old points can be deleted. The algorithms we propose for insertion and deletion use lifted flips as the only topological operation and are based on a geometrical interpretation of the history data structure. Our approach can be included in the class of incremental algorithms proposed by Clarkson, Melhorn and Seidel [4].

The rest of the paper is organized as follows: in Section 2 definitions and important properties are given for regular triangulations. Next, in Section 3, we revise the incremental algorithm for inserting a point in a d-dimensional regular triangulation, and in Section 4 the history data structure used by this algorithm is interpreted geometrically. Our proposal is introduced in Section 5, including the history data structure we use, and the two basic operations on this structure, the rotation and the lifted flip. The algorithm for inserting a new point in a regular triangulation is presented in Section 6, and the algorithm for point deletion is described Section 7. We also give a counterexample which proves that unsorted flipping does not work in deletion. Finally, results are shown and conclusions are given.

2 Regular triangulations

Given a finite set of points $S = \{p_1, p_2, \ldots, p_n\} \subset \mathbb{R}^d$, a triangulation $\mathcal{T}$ of S is a decomposition of the region bounded by the convex hull of S, $CH(S)$, into non-overlapping d-simplices Δ_d where the vertices of the simplices are points in S. In addition, if two simplices intersect, their intersection is a face of both. Given a set $t \subset \mathbb{R}^d$ of $d+1$ affinely independent points we denote by $\Delta_d(t)$ the d-dimensional simplex with vertices in t.

The Delaunay triangulations $\mathcal{DT}(S)$ are a class of triangulations which play an important role in a variety of different disciplines of science. In Delaunay triangulations, each simplex satisfies the *empty-sphere* property, that means that the circumsphere of a simplex does not contain any point of S in its interior. The Delaunay triangulation $\mathcal{DT}(S)$ is the dual of the Voronoi diagram, which is a fundamental tool expressing the proximity of geometric objects. Another way to think on Delaunay triangulations is

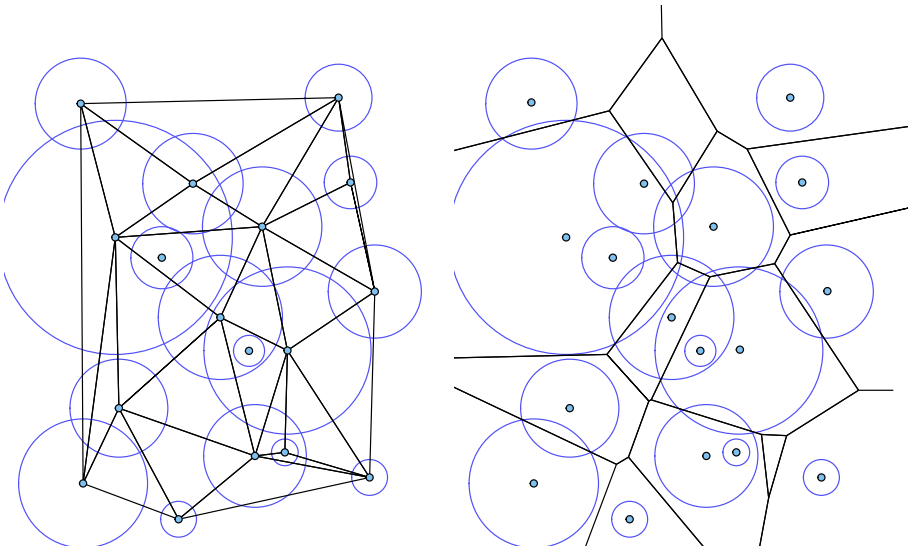

Figure 1: *Left*, a set of weighted points in $\mathbb{R}^2$ and its regular triangulation; *right*, the Voronoi diagram of the set. Blue circles represent the spheres $C_\omega(p)$.

through their relationship with convex hulls in $\mathbb{R}^{d+1}$. Using a standard transformation that lifts the sites of S to the paraboloid of revolution $U_{d+1} : x_{d+1} = x_1^2 + x_2^2 + \ldots + x_d^2$ in $\mathbb{R}^{d+1}$, $\mathcal{DT}(S)$ can be constructed computing the (lower) convex hull of the transformed sites [7].

Regular triangulations are a generalization of Delaunay triangulations that maintain both their relationship with convex hulls and with Voronoi diagrams. In this case, a real valued weight ω_p is assigned to each point p of $S \subset \mathbb{R}^d$. This weight can be interpreted as a sphere $C_\omega(p)$ with center p and radius $\sqrt{\omega_p}$. Usually, as we do, positive weights are assumed, although there is no theoretical inconvenient in taking spheres with negative radius. Figure 1 shows an example of a regular triangulation of a set of weighted points in the plane.

In case of regular triangulations, their relationship with convex hulls uses the weight in the lifting step. Given a point $p \in S$ with coordinates $(x_1, x_2, \ldots, x_d)$ and associated weight ω_p, we define the lifted point (with respect to regular triangulations) to be $p^+ = (x_1, x_2, \ldots, x_d, x_1^2 + x_2^2 + \ldots + x_d^2 - \omega_p) \subset \mathbb{R}^{d+1}$. A lifted set of points C^+ is the set of lifted points in C, for example $S^+ = \{p^+ \mid p \in S\}$. Notice that if the weight is different from zero, the lifted point p^+ lies off the paraboloid in $\mathbb{R}^{d+1}$. In this case, the projection of the lower facets of $CH(S^+)$ gives the regular triangulation $\mathcal{RT}(S)$. We note by $LCH(S^+)$ the lower convex hull of the set of lifted points S^+. A point p may not be incident to any facet of $LCH(S^+)$, but interior to the hull; in this case, p is not a vertex of $\mathcal{RT}(S)$ and we say that p is a *redundant point* in the triangulation.

Regular triangulations can also be seen as the dual, in a graph-theoretical sense, of Voronoi decompositions. In this case, the power distance, $\|\cdot\|_w$, replaces the Euclidean distance, $\|\cdot\|$ [16]. The power distance from a point $q \in \mathbb{R}^d$ to a weighted point p is

defined as

$$||pq||_\omega = ||pq||^2 - \omega_p$$

This distance can be interpreted geometrically as the square of the length of a segment from q to the point r in $C_\omega(p)$ such that qr is tangent to $C_\omega(p)$. The power distance defines a Voronoi diagram of a set of weighted points, which is called the power Voronoi diagram, and its dual is the regular triangulation of the given weighted points (see Figure 1). Redundant points are the ones whose Voronoi region is empty, that is, points $p \in S$ such that any point q of $\mathbb{R}^d$ is closer to another point $p' \in S$, different from p, using the power distance.

3 The d-dimensional algorithm

In this section we revise the incremental algorithm for adding a weighted point in a regular triangulation $\mathcal{RT}(S)$ as given in [8]. Since a precondition for the algorithm to work is that the point must belong to the convex hull of the triangulated region, an usual implementation practice is to initialize the triangulation with an extra d-simplex that contains the whole set of points S (see Section 5.1). The algorithm that given a regular triangulation $\mathcal{T}$ of a set of points S in d dimensions obtains the regular triangulation of the augmented set $\mathcal{RT}(S \cup \{x\})$ is the following:

Locate the simplex $\Delta_d(t) \in \mathcal{T}$ that contains x
If *x is not locally redundant in t* **then**
  Flip $t \cup \{x\}$
  While *there exist locally non-regular facets* **do**
    Find a locally non-regular facet Δ_{d-1} that is flippable
    Flip Δ_{d-1}
  endWhile
endIf

In this algorithm, $\Delta_d(t)$ is the simplex that contains the vertex x, and t is the set of $d+1$ vertices of this simplex. A *facet* Δ_{d-1} of $\Delta_d(t)$ is a $(d-1)$-simplex with vertices in t. The algorithm only has to test for non-regular facets opposite to the vertex x being inserted.

Given a set t of $d+2$ weighted points in $\mathbb{R}^d$, there are exactly two ways to triangulate t; one of the two triangulations is the regular one, the other is not regular. The regular way coincides with the vertical projection of the lower facets of the $(d+1)$-simplex in $\mathbb{R}^{d+1}$. A *flip* is the topological operation that replaces one triangulation of t with the other. Flips can be classified into different types, depending on the number of d-simplices before and after the operation; we denote by $x \to y$ a flip that replaces x d-simplices by y d-simplices. In $\mathbb{R}^2$ there are three different types of flips, $1 \to 3$, $2 \to 2$ and $3 \to 1$; in $\mathbb{R}^3$ there are four (see Figure 2). A $1 \to (d+1)$ flip is an operation that replaces a d-simplex plus an internal vertex by d neighbor d-simplices incident to the vertex.

The first step when a new point x is added into $\mathcal{T}$, is to locate the d-simplex that contains it. Then, the algorithm tests whether the point is redundant in the regular

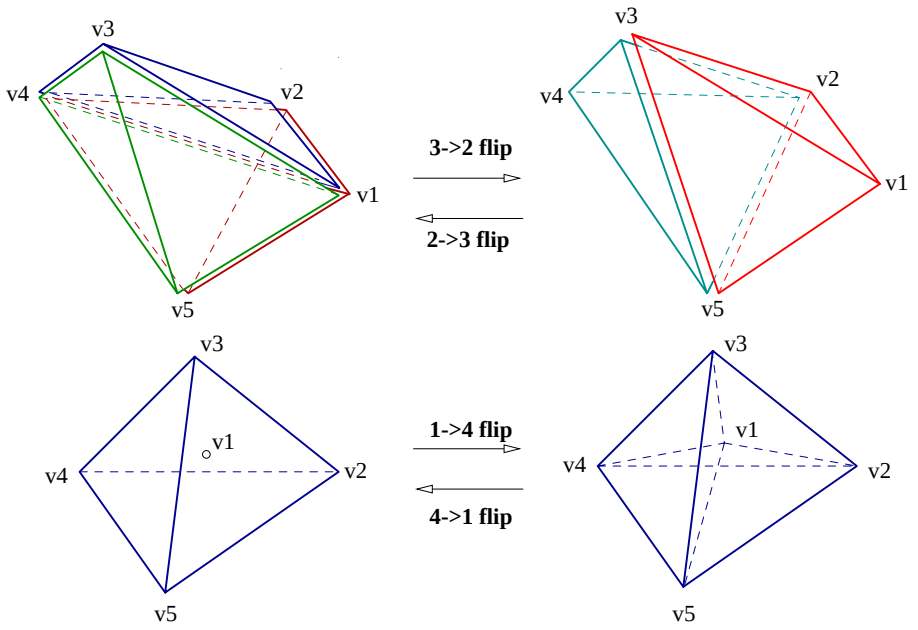

Figure 2: The four types of flips in 3 dimensions.

triangulation (the point will be redundant in $\mathcal{RT}(S \cup \{x\})$ if and only if it is locally redundant in t). In this case, it is discarded, otherwise a $1 \rightarrow (d+1)$ flip is performed, connecting all the vertices t to x. The algorithm proceeds by flipping locally non regular $(d-1)$-simplices that are flippable until none remains. To flip a facet means to perform the flip of the set of vertices of the two simplices incident to the facet. The resulting triangulation is the regular triangulation of $S \cup \{x\}$.

We still have to define when a facet Δ_{d-1} is flippable. Let $V = v_1 \ldots v_d$ be the d $(d-2)$-simplices of Δ_{d-1}, and $\Delta_d{}'$, $\Delta_d{}''$ the two simplices incident to Δ_{d-1}. Call v_i *convex* if there is an hyperplane that contains it and both $\Delta_d{}'$ and $\Delta_d{}''$ lie on the same side of this hyperplane; otherwise call v_i *reflex*. According to [8], Δ_{d-1} is *flippable* if and only if all reflex $(d-2)$-simplices of V have degree three, that is, each is incident to exactly three $(d-1)$-simplices.

The usual implementations of the above algorithm use a stack (or a list) of non-regular facets to be considered for flipping, so the algorithm converges when the stack is empty. Since a flip can imply more than one (non-regular) facet incident to x, each time a facet is flipped all the other facets that disappear have to be removed from the stack; or either each time is popped from the stack, it has to be checked that the facet still belongs to the current triangulation.

The first step of the algorithm, vertex location, is a crucial one because it can determine the average complexity of the algorithm. The most common way of performing it is searching with the help of an additional data structure, usually a historical structure (see Section 4). In this way, the expected time for the location of the n points of set S,

which coincides with the expected time for the whole algorithm, is $O(n \log n + n^{\lceil d/2 \rceil})$ for a d-dimensional regular triangulation of n randomly placed points [8]. In a recent paper, Mücke, Saias and Zhu showed that vertex location can also be performed "marching" on the self triangulation, without the help of an additional data structure. Locating a single vertex using this method takes expected time $O(n^{1/(d+1)})$ in a Delaunay triangulation of n randomly placed points in $d = 3$ dimensions [15].

4 Geometric interpretation of the history data structure

The most widely used auxiliary data structure for locating points on a triangulation is a history structure, which stores the sequence of flips performed when adding new point into the triangulation. Guibas, Knuth and Sharir see it as a set of layers of triangulations (in the author's words, triangulations are stored "one on top of another" [12]); Edelsbrunner and Shah [8], and also Facello [10] store it as a history DAG; and Boissonat, Teillaud and Devillers use a modified version of this data structure, called the Delaunay tree [2, 5].

Originally these data structures were only used for vertex location, but later, since when deleting a vertex from the triangulation the auxiliary data structure had to be updated accordingly, it was seen that the self structure was useful for the vertex removal operation. The use of a history data structure was generalized by Clarkson, Mehlhorn and Seidel as a general scheme for randomized dynamic algorithms [4]. Specifically, the authors showed how this history structure can be used to maintain the convex hull of a dynamic set of points. Recently, Edelsbrunner and Waupotitsch proposed to use the history structure for mantaining simplicial grids that vary under density requirements [9].

The correspondence of flips in $\mathbb{R}^d$ with the vertical projection of a $(d+1)$-simplex from $\mathbb{R}^{d+1}$ into one less dimension gives a well-known geometric interpretation of the incremental flipping algorithm presented in previous section. We start with the regular triangulation of S, i.e., the lower convex hull of S^+, $LCH(S^+)$. Once the simplex $\Delta_d(t)$ that contains x is located, the test for local redundancy means to classify x^+ with respect to supporting hyperplane π through vertices in t^+. If x^+ lies above π, it is a redundant point, otherwise it lies outside $LCH(S^+)$ and has to be added into the regular triangulation accordingly.

The first flip connects x^+ to the vertices in t, thus a new $(d+1)$-simplex is created that joins x^+ to $LCH(S^+)$. The other flips also create $(d+1)$-simplices incident to x^+, and must only be performed if a lower facet Δ_{d-1} surrounding x^+ is non-convex in $\mathbb{R}^{d+1}$, that is to say, only if the facet is locally non-regular in $\mathbb{R}^d$. A point that was on the boundary of $LCH(S^+)$ can be left in the interior of the new lower convex hull by a $(d+1) \rightarrow 1$ flip, and then the point becomes redundant. The process is iterated until no flips can be performed, and the new convex hull is obtained.

This correspondence between regular triangulations and convex hulls also allows a simple geometric interpretation of the history data structure: the sequence of flips is nothing else than the historical triangulation of the interior of the lower convex hull

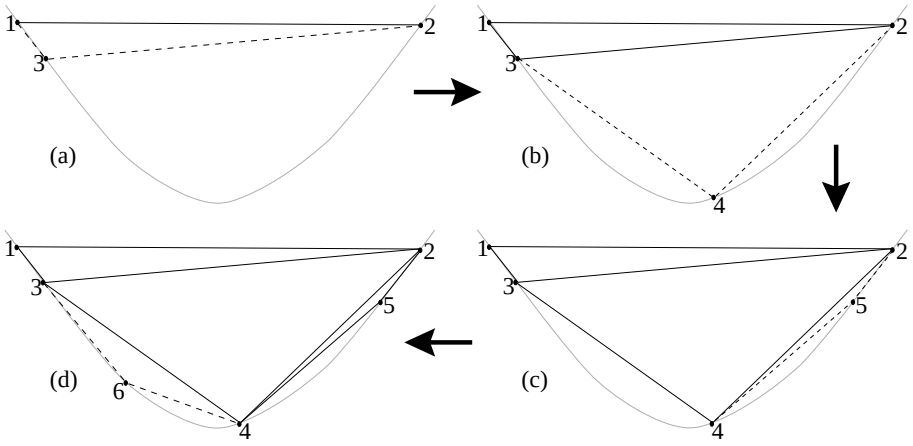

Figure 3: The history data structure interpreted as the triangulation in one more dimension. The figure shows four stages of the incremental computation of the Delaunay triangulation of a set of points in one dimension, mapped into the parabola in $\mathbb{R}^2$. Each time a point is inserted, a new triangle is created (dashed lines) and the new lower convex hull is obtained.

mapped in one more dimension.

Figure 3 illustrates this interpretation for a set of one-dimensional weighted points (for better comprehension), thus the history data structure is two-dimensional and the lower convex hull is a convex polygonal with endpoints at the lifted point set. Since the unweighted Delaunay triangulation is being computed, all the points have the same weight, therefore they lie on the same parabola. The figure shows four stages of creation of $\mathcal{DT}(S)$. Initially, the triangulation contains only points 1 and 2, thus the Delaunay triangulation is composed of a segment that joins these two points (Figure 3a). When a new point is inserted (Figures 3b–d), a $1 \rightarrow 2$ flip is needed that joins the vertex to the old lower convex hull, obtaining the new lower convex hull. Each time a flip is performed, a new triangle is created. The set of these triangles constitutes the history data structure.

In Figure 4, point 10 is going to be inserted. The first step consists on locating the simplex where it is included. This is done starting from the topmost triangle and descending (marching downwards) through the historical triangulation until we run to the exterior (blue lines in the figure). In the example, the algorithm for updating $\mathcal{DT}(S)$ will add the triangle with vertices $8, 4, 10$ to the historical triangulation (performing a $1 \rightarrow 2$ flip in one dimension). Since the Delaunay triangulation is being builded and we are working in one dimension, in this very particular case no other triangles will be added in the historical triangulation (no additional flips are necessary). If, instead, the triangulation was regular, points could have different weights, thus they would not lie on the same parabola. In this case, the vertex location step may find that the point is interior to the hull, i.e., it is a redundant point in $\mathcal{RT}(S)$, so it does not need to be added. If the point is not redundant, a series of $2 \rightarrow 1$ flips could be needed (imagine,

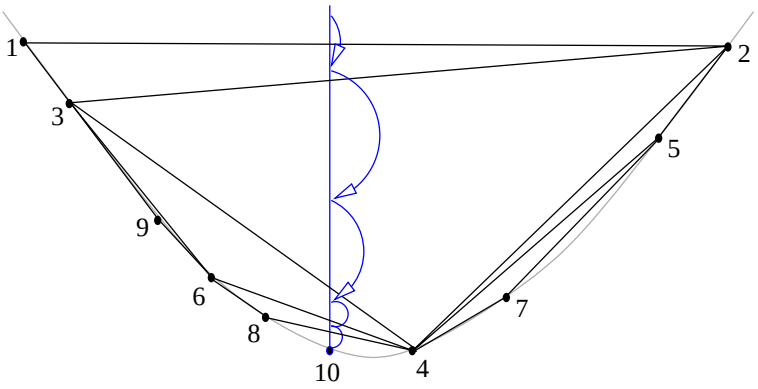

Figure 4: Point 10 is being inserted in the triangulation. Locating the simplex that contains it is equivalent to march downwards through the history triangulation until the surface of the hull is reached.

for example, that tenth point lies below the parabola, that is, it has weight greater than the rest of the points). Since these flips would add new triangles to the historical triangulation, some points in the old triangulation can become redundant (i.e. interior to the lower convex hull).

5 Algorithms for dynamic regular triangulations

We have described the incremental method for adding a new point into a regular triangulation. There also exist several methods for removing a single point from a Delaunay triangulation, but the problem when removing points from regular triangulations is that redundant points can become vertices of the triangulations (interior vertices that become part of the hull). In fact, algorithm proposed by Devillers in [6] for deletion in Delaunay triangulations also works for regular triangulations provided that no redundant points exist. The incremental algorithm of Section 3 does not store points if they are redundant in the current triangulation, therefore these points cannot be easily taken into account later when a point is removed. (However, notice that some redundant points will be kept in the history structure, if they were non-redundant in a previous stage.)

Our proposal uses the geometric interpretation seen on the previous section not only for adding a new vertex in $\mathcal{RT}(S)$, but also to delete an existing one, managing a fully dynamic triangulation. The only topological operations used by addition and deletion algorithms are lifted flips (see below), and no auxiliary data structure is used apart from the historical triangulation. Although, for simplicity, the algorithms we give are described for a set of weighted points in $\mathbb{R}^2$, they can be straightforward generalized to any dimension.

The main difference between our proposal and previous approaches comes from the use of a new history data structure. In our data structure, all the points, redundant or

not, are always stored in the same way. This solution allows us to unify the treatment for inserting and deleting points, regardless of they are redundant or a vertex of the current regular triangulation, therefore rather simple algorithms are obtained. In [4] an alternative way is proposed, storing for each point a pointer to a $(d+1)$-simplex that contains it, and for each simplex a list of interior redundant points. However, this solution implies modifying the pointers and splitting the lists whenever topological changes withdraw simplices with non-empty associated lists (which happens during the deletion of vertices).

5.1 A new history data structure

The data structure we propose is simply the historical tetrahedrization of the set of lifted points S^+. A point is a tuple containing its lifted coordinates in $\mathbb{R}^3$, and a label (an integer) that will be used for deletion. The historical data structure is stored as a tetrahedrization spanning all the points in S^+. Each tetrahedron stores four pinters to its vertices and four other pointers to the neighbor tetrahedra.

As mentioned, since the insertion algorithm requires the new point to be interior to the convex hull of the triangulated region, we initialize the triangulation as a single artificial tetrahedron $\Delta_{d+1}(SV)$ that contains the whole set of points S. Assuming that all vertices in S have positive weigth, the four vertices $SV = \{sv_1, sv_2, sv_3, sv_4\}$ of this initial simplex, henceforth referred as super-vertices, can be chosen to have coordinates and weights

$$\begin{aligned} sv_1^+ &= (0, +\infty)\ ,\ weight(sv_1) = 0\ , \\ sv_2^+ &= (-\infty, -\infty)\ ,\ weight(sv_2) = 0\ , \\ sv_3^+ &= (+\infty, -\infty)\ ,\ weight(sv_3) = 0\ , \\ sv_4^+ &= (0, 0)\ ,\ weight(sv_4) = +\infty\ . \end{aligned}$$

In practice, the symbol ∞ is replaced by a large enough number. Notice that, unlike other algorithms for Delaunay or regular triangulations, we use a fourth super-vertex with infinite weight, sv_4. This last super-vertex is the key that allows to simplify the insertion and deletion algorithms, not having to distinguish the case when the point being inserted is redundant from the case it is a vertex of $\mathcal{RT}(S)$. The augmented set of points is $S^* = S^+ \cup SV^+$, and the history data structure, i.e., the tetrahedrization of S^*, is notated $\mathcal{H}(S^*)$.

In the history structure we propose each vertex has a label associated. This label is an index which indicates the order that the points have to be inserted in $\mathcal{H}(S^*)$ in such a way that the lower convex hull of S increases. We define $S_k \subset S$ to be the set of vertices with label less or equal to k,

$$S_k = \{v \in S \mid label(v) \leq k\}.$$

Notice that this label does not reflect the real order that the points have been inserted in practice. The history data structure stores a spanning tetrahedrization of the interior of the convex hull of $S^+ \cup \{sv_1^+, sv_2^+, sv_3^+\}$. We state this using the following property:

Property 1. *Let $\mathcal{H}(S^*)$ be the history data structure that stores the tetrahedrization of points in S^*. Let $\mathcal{T}_k \subset \mathcal{H}(S^*)$ be the set of simplices with vertices in S_k^+. Then, the lower hull of $\mathcal{T}_k$ is the lower convex hull of S_k^+, $LCH(S_k^+)$, and therefore the vertical projection of $\mathcal{T}_k$ gives $\mathcal{RT}(S_k)$.*

In fact, Property 1 is the invariant condition that ensures the correctness of the global algorithm for a dynamic set of points. Therefore, the algorithms we are going to develop will ensure that this condition still holds after insertions and deletions. To achieve this, we assign label $-\infty$ to three super-vertices, and the last super-vertex, sv_4, has assigned label $+\infty$ (in practice, the first three supervertex can have negative labels, and sv_4 can have label $N+1$, being N the maximum label of any vertex in S). In this way, we "simulate" that vertices in S are inserted after the three first super-vertices and before sv_4.

A straightforward consequence of Property 1 is that vertices in S connected to sv_4 are the vertices of $\mathcal{RT}(S)$, and that the opposite faces of tetrahedra incident to sv_4 are the triangles of $\mathcal{RT}(S)$. Thus, obtaining the triangles of the regular triangulation is immediate: triangles in $\mathcal{RT}(S)$ are the vertical projection of faces of $\mathcal{H}(S^*)$ with endpoints in S and opposite to sv_4. In fact, for efficiency, the relationship between neighbor triangles in $\mathcal{RT}(S)$ can also be maintained, although it can be easily derived from the neighbor pointers between tetrahedra incident to sv_4.

As it has been explained, vertex location of a point x in the tetrahedrization is equivalent to downwards marching from the topmost tetrahedron (the only one incident to sv_1, sv_2 and sv_3). Once the tetrahedron Δ_3 that contains x is reached, if it is incident to sv_4, point x will become a vertex of $\mathcal{RT}(S \cup \{x\})$, otherwise x is a redundant point. The only geometric test necessary for vertex location by downward marching can be performed by vertical projection in $\mathbb{R}^2$, because it consists on classifying a vertical line l that cuts an upper face of a tetrahedron with respect to its lower faces, selecting the one which is cut by l. The search will stop when x is found to lie above the supporting plane of a lower facet.

5.2 The rotation operation

Notice that the label associated to each vertex depends on the position of the lifted point and on the insertion order of the vertex into the set S. Thus, given two vertices v_k and v_{k+1} with consecutive labels and such that $v_k \notin CH(S_{k+1}^+ - \{v_k\}^+)$, then exchanging the labels of v_k and v_{k+1} gives a different data structure of the same set of weighted vertices S, which is also valid (i.e. it still fulfills Property 1). Although the set of points remains the same, the attached labels are different and, thus, the historical triangulation also has to be modified.

This allows to define an operation, called *rotation* of the vertices v_k and v_{k+1} in $\mathcal{H}(S^*)$ [9]. The rotation of v_k and v_{k+1} changes $\mathcal{H}(S^*)$ into $\mathcal{H}(S'^*)$, where S' is the same set of weighted points of S^* except that points v_k and v_{k+1} have permuted their labels (see Figure 5).

A rotation only implies to rearrange part of the region comprised between $LCH(S_{k-1}^+)$ and $LCH(S_{k+1}^+)$. We define $V(S,x,y)$ to be the set of faces in $LCH(S^+)$ visible

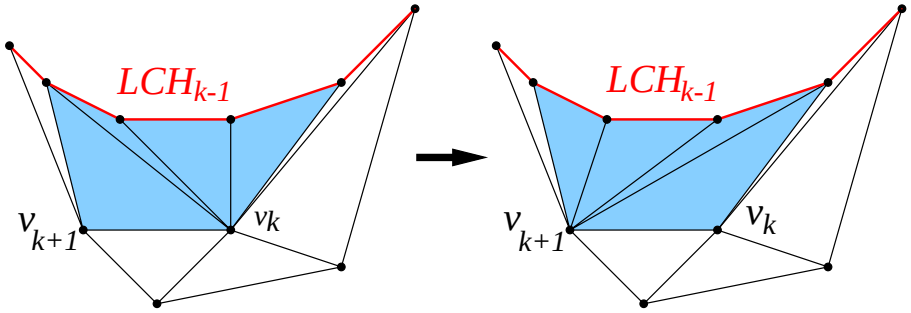

Figure 5: Representation of a rotation of vertices v_k and v_{k+1} in $\mathbb{R}^{d+1} = \mathbb{R}^2$. Shaded region is the one that needs to be modified.

from vertices x and from y. Then, the new tetrahedrization $\mathcal{H}(S'^*)$ is obtained in the folowing way:

(i) Removing the tetrahedra incident to v_k^+ and to a face in $V(S_{k-1}, v_k, v_{k+1})$; let H_k be the set of these tetrahedra.
(ii) Removing the tetrahedra incident to v_{k+1}^+ to any face of H_k.
(iii) Inserting new tetrahedra that connect v_{k+1}^+ to $V(S_{k-1}, v_k, v_{k+1})$; let G_{k+1} be these new tetrahedra.
(iv) Inserting tetrahedra that connect v_k^+ to new faces of the lower convex hull of G_{k+1}.

5.3 Lifted flips

We have yet seen that flips in $\mathbb{R}^d$ can be interpreted as the vertical projection of a $(d+1)$-simplex from $\mathbb{R}^{d+1}$. In this section we introduce another interpretation of a flip in $\mathbb{R}^d$ as a flip lifted in $\mathbb{R}^{d+1}$ seen from a vertex. This notation, and the other additional notation given here, is used in the following sections to understand how the insertion and deletion algorithms work.

A flip in $\mathbb{R}^d$ implies a set t of $d+2$ vertices and $d+2$ d-simplices with vertices in t. To interpret this flip in one more dimension, we need to lift vertices in t and to add an additional vertex v^+ independent from the points in t^+. This vertex will be called the *observer vertex*. Connecting the vertices of t^+ to v^+, each d-simplex gives a $(d+1)$-simplex. Hence, flipping t in $\mathbb{R}^d$ can be seen in $\mathbb{R}^{d+1}$ as an operation that replaces k $(d+1)$-simplices with $d+2-k$ $(d+1)$-simplices. If the $(d+1)$-simplex incident to the vertices in t^+ is also considered, this operation is topologically equivalent to a $(d+1)$-dimensional flip.

However, notice that the operation in $\mathbb{R}^{d+1}$ obtained adding the observer vertex v^+ to a flip in $\mathbb{R}^d$ is not always a valid flip, because in some cases the set of simplices in $\mathbb{R}^{d+1}$ does not give a non-overlapping triangulation. We next define a lifted flip, which is the topological operation used by our algorithms.

Definition 1 (Lifted flip). *Let t be a set of weighted points in $\mathbb{R}^d$. A flip of t replaces*

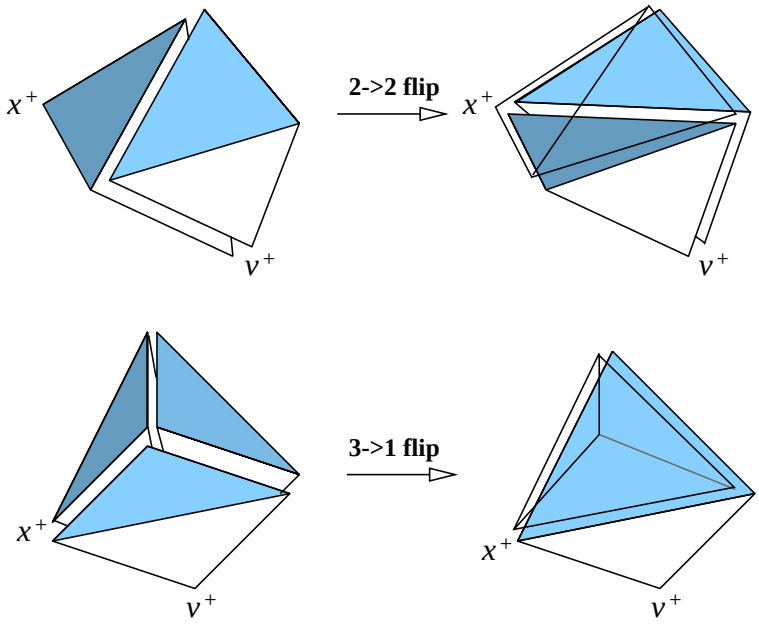

Figure 6: Lifted flips in $\mathbb{R}^2$ seen from vertex v^+ interpreted as flips in $\mathbb{R}^3$. In the left configurations, one of the tetrahedra – the topmost – is drawn transparent.

the k d-simplices $\Delta(r_1), \ldots, \Delta(r_k)$ with d-simplices $\Delta(r_{k+1}), \ldots, \Delta(r_{d+2})$. The corresponding lifted flip seen from v^+ *is the topological operation that replaces the k $(d+1)$-simplices $\Delta(r_1^+ \cup \{v^+\}), \ldots, \Delta(r_k^+ \cup \{v^+\})$ with $\Delta(r_{k+1}^+ \cup \{v^+\}), \ldots, \Delta(r_{d+2}^+ \cup \{v^+\})$ plus the additional simplex $\Delta(t^+)$.*

The definition of a lifted flip gives a condition for a facet to be flippable in $\mathbb{R}^{d+1}$, but we restrict to the case $d = 2$ for simplicity.

Definition 2 (L-flippability). *Let t be a set of four weighted points in $\mathbb{R}^2$, and let v^+ be the observer vertex. A facet in $\mathbb{R}^3$ incident to v^+ and to two points $a^+, b^+ \in t^+$ is* l-flippable *if and only if*

(i) edge ab is flippable (in $\mathbb{R}^2$), and
(ii) ab is non-regular.

Figure 6 shows the two types of flips in $\mathbb{R}^2$ interpreted as lifted flips seen from vertex v^+ which are l-flippable. In the figure, blue faces are obtained lifting points to $\mathbb{R}^3$, and transparent tetrahedra are the additional ones created from all the original points. Notice that the position of the observer vertex v^+ is choosen in the figure so that lifted flips are in fact flips in $\mathbb{R}^3$ (the upper case is a $2 \to 3$ flip and the lower case is a $3 \to 2$ flip). However, in general not all lifted flips correspond to valid flips, although they are topologically correct operations.

6 Point insertion

The algorithm we give for inserting a new point in a regular triangulation follows the scheme of the incremental approach of [8]. Nevertheless, our proposal works entirely in $\mathbb{R}^{d+1}$, and we use $\mathcal{H}(S^*)$ as the only geometric data structure instead of storing the current triangulation in $\mathbb{R}^d$ and updating the history DAG adequately. Furthermore, any point, redundant or not, is a vertex of the tetrahedrization (i.e., $\mathcal{H}(S^*)$ is a spanning tetrahedrization of S).

The point to be inserted will be called x, and the initial point set of points is $S = \{v_{i_1}, \ldots, v_{i_n}\}$, where i_k is the label of v_{i_k} and $i_k \leq N$. The insertion algorithm works in the following way: if x is not redundant in S, it is added in $\mathcal{H}(S^*)$ as the vertex with maximum label (except for sv_4); otherwise, it is added assigning to it the highest possible label, which is the maximum label of the vertices of the tetrahedron that contains x. We define $maxlabel(\Delta_2)$ to be the maximum label of vertices incident to the facet Δ_2, that is

$$maxlabel(\Delta_2) = \max\{label(v) \mid v \in t\}$$

being t the set of vertices of the two tetrahedra incident to Δ_2. Notice that $maxlabel(\Delta_2) = j$ implies that v_j is a vertex of at least one of the two tetrahedra incident to Δ_2.

The pseudocode algorithm that inserts a new point x in $\mathcal{H}(S^*)$ is the following:

March downward to locate the tetrahedron $\Delta_3(t) \in \mathcal{H}(S^*)$
that contains x^+
Let $t = \{v_a, v_b, v_c, v_d\}$
$j := \max(a, b, c, d)$
$L :=$ *EmptyList()*
Flip $t^+ \cup \{x^+\}$
Add to L *the three faces of* $\Delta_3(t)$ *incident to* v_j^+
While L *is not empty* **do**
  GetFirst(L, Δ_2)
  If Δ_2 *is l-flippable seen from* v_j^+ **then**
    Perform the lifted flip of Δ_2 *seen from* v_j^+
    Update L *with new faces* Δ *such that* $maxlabel(\Delta) = j$
  endIf
endWhile
If $j = +\infty$ **then**
  SetLabel$(x, N + 1)$
else
  Increment the label of vertices in $S - S_{j-1}$
  SetLabel(x, j)
endIf

We use a list L of faces of tetrahedra (triangles) that are pending to be flipped. All triangles in L are incident to v_j^+ and are facets of tetrahedra opposite to x^+. The call to procedure $GetFirst$ consults and removes the first element from list L, saving it as Δ_2. The first flip is a $1 \rightarrow 4$ flip that adds point x into the tetrahedrization, and at the

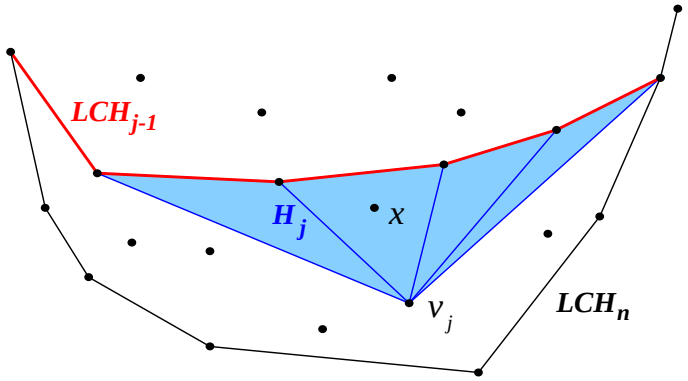

Figure 7: Insertion of a redundant point x into a regular triangulation in $\mathbb{R}^{d+1} = \mathbb{R}^2$ (super-vertices are not represented).

beginning the list L will contain three of four faces of $\Delta_3(t)$ opposite to x^+. The other flips are lifted flips seen from vertex v_j^+. These lifted flips change things from one of the configurations in the left of Figure 6 to the corresponding configuration on its right, being v_j^+ the observer vertex. When L is updated, a test is made to ensure that new faces are not included in L twice, and the faces that have been deleted by the flip are removed from L if they were in it. This operation can be efficiently performed if each facet stores a pointer to its position in the list.

In the case the point x is non-redundant in $\mathcal{RT}(S)$, the algorithms works as the one given in Section 3 but performing flips in $\mathbb{R}^{d+1}$. Indeed, in this case the tetrahedron $\Delta_3(t)$ that contains x^+ is incident to sv_4^+, thus j is assigned to $+\infty$ (the label of sv_4), since $v_j = sv_4$. The three faces added in L at the beginning are the lower faces of $\Delta_3(t)$. Now, all we have to do is to take notice that successive lifted flips performed by the algorithm can be understood as $2\to2$ and $3\to1$ flips in $\mathbb{R}^2$ seen from sv_4^+. Each flip adds a new tetrahedron to the lower convex hull of S until the new regular triangulation is obtained. Indeed, the condition for l-flippability is a test for lower faces incident to x^+ implied in the lifted flips (see Section 5.3). Finally, the algorithm assigns label $N+1$ to vertex x, i.e. x will be labeled as the vertex inserted after all the vertices in S except for sv_4 (in practice, N can be a counter for the number of insertions).

In the other case, when x is redundant in $\mathcal{H}(S^*)$, the vertex labeling is modified so that point x can also be stored as a vertex of the tetrahedrization. In this case, the algorithm performs a series of lifted flips so that the resulting tetrahedrization is the one where x was inserted just before vertex v_j. This is equivalent to considering that x is added to $\mathcal{RT}(S_{j-1})$. First, the algorithm does a $1\to4$ flip that adds x inside the tetrahedrization. Subsequent lifted flips can be interpreted as flips in $\mathbb{R}^2$ seen from v_j^+ that add new tetrahedra to $LCH(S_{j-1}^+)$ until the regular triangulation of $S_{j-1}^+ \cup \{x\}$ is obtained as the lower projection of the new tetrahedrization if only tetrahedra with vertices on this set were considered. The final part of the algorithm simply shifts labels for vertices in $S - S_{j-1}$ so that x can be labeled as j.

We can interpret an insertion of a new vertex into $\mathcal{H}(S^*)$ as a special case of a

rotation: inserting x is the same than locating the tetrahedron $\Delta_3(t)$ that contains x, performing a $1 \rightarrow 4$ flip, and rotating x and v_j. Figure 7 represents the insertion process for a set of one-dimensional weighted points lifted to $\mathbb{R}^2$. In the figure, LCH_n is the lower convex hull of S^+, and LCH_{j-1} is the lower convex hull of S^+_{j-1}. We call H_j the set of simplices connecting v_j^+ to $LCH(S^+_{j-1})$ (blue region in Figure 7). Obviously, the simplex $\Delta_d(t)$ that contains x is one of H_j. The following two lemmas characterize the simplices in $\mathcal{H}(S^*)$ that must be removed and the ones that need to be added in order the new tetrahedrization including x^+ fulfills Property 1.

Lemma 1. *A tetrahedron Δ_3 in $\mathcal{H}(S^*)$ will not be in $\mathcal{HS}(S^* \cup \{x^+\})$ if and only if $\Delta_3 \in H_j$ and the opposite face to v_j^+ is one of $V(S_{j-1}, x, v_j)$.*

Proof. First, notice that since $LCH(S^+_{j-1})$ is convex and x^+ is interior to H_j, any face of $LCH(S^+_{j-1})$ visible from x^+ is also visible from v_j^+, hence only part of H_j has to be modified. Since in the final result x will be assigned the label just before the label of v_j, any face in $V(S_{j-1}, x, v_j)$ will be joined to x^+, thus they cannot be joined to v_j^+. □

Lemma 2. *Tetrahedra in $\mathcal{HS}(S^* \cup \{x^+\})$ that were not in $\mathcal{H}(S^*)$ are:*

(a) Tetrahedra joining x^+ to a face in $V(S_{j-1}, x, v_j)$; call G_x the set of these tetrahedra.
(b) Tetrahedra joining v_j^+ to the faces of $LCH(G_x)$.

Proof. (a) The tetrahedra of G_x are the ones needed for $LCH(S^+_{j-1} \cup \{x^+\})$ to be faces of $\mathcal{HS}(S^* \cup \{x^+\})$. Recall that facets of $LCH(S^+_{j-1})$ visible from x^+ are also visible from v_j^+.

(b) To complete $\mathcal{HS}(S^* \cup \{x^+\})$, we need tetrahedra filling the gap between $LCH(S^+_{j-1} \cup \{x\}^+)$ and $LCH(S^+_j \cup \{x^+\})$ that were not in $\mathcal{H}(S^*)$. These tetrahedra join the vertex v_j^+ with the faces of tetrahedra in G_x visible from v_j^+, that is, faces of $LCH(G_x)$. Notice that all these lower faces are visible from v_j^+.

□

We are now ready to prove that insertion algorithm updates the history data structure as required.

Theorem 1. *The algorithm for insertion obtains $\mathcal{HS}(S^* \cup \{x^+\})$ fulfilling Property 1.*

Proof. Since the algorithm only tests for flippability of faces with maximum label j, it will only perform lifted flips of tetrahedra incident to v_j^+ between $LCH(S^+_j)$ and $LCH(S^+_{j-1})$, that is, tetrahedra belonging to H_j. Notice that the lifted flips and flippability conditions are all performed being v_j^+ the observer vertex. If edges incident to v_j^+ are not considered, a lifted flip of a l-flippable face is equivalent to a flip of a flippable and locally non-regular edge when projected to $\mathbb{R}^2$ (see Section 5.3). Therefore, the insertion algorithm reduces to the incremental algorithm given in Section 3 if only vertices in S_{j-1} are considered. This implies that tetrahedra of G_x (those of

Lemma 2(a)) are created, and also that tetrahedra of H_j not visible from x remain unchanged. To finish the proof it is sufficient to see that the insertion algorithm in $\mathbb{R}^3$ creates a valid tetrahedrization (since this implies that tetrahedra of Lemma 1 will be deleted and tetrahedra of Lemma 2(b) will be created).

A lifted flip obtains a new topologically correct structure (see Section 5.3). Hence, when the insertion algorithm finishes a topologically valid tetrahedrization is obtained. Since a lifted flip seen from v_j^+ corresponds to a flip in $\mathbb{R}^2$, all tetrahedra created by the insertion algorithm either belong to G_x or they are incident to v_j^+. Due to the way the incremental lifted flipping works, when the algorithm finishes all tetrahedra in G_x form a correct tetrahedrization joining v_j^+ to $LCH(S_{j-1}^+)$. Besides, the topological correctness of the process assures that at the end v_j^+ will be connected to faces in $LCH(G_x)$. □

7 Point deletion

The algorithm we propose for removing a point from a regular triangulation is inspired in the algorithm for maintaining dynamic convex hulls given in [4], and uses the same simple idea for deleting a vertex x from a regular triangulation: to reconstruct the tetrahedrization (the history in one more dimension) so that x was never inserted.

Note also that the algorithmic scheme followed by our deletion procedure is in some way similar to the one in [6]. However, our algorithm works in $\mathbb{R}^{d+1}$ and uses a different sorting criterion for the flips, which makes it a comletely different approach.

The algorithm for deleting vertex x from $\mathcal{H}(S^*)$ in pseudocode is the following:

$Q :=$ *EmptyPriorityQueue()*
Add to Q *all the facets incident to* x^+
While Q *has more than 4 faces* **do**
  GetFirst(Q, Δ_2)
  Perform the lifted flip of Δ_2 *seen from* x^+
  Add to Q *new facets incident to* x^+
  Update facets in Q *whose sorting index has been modified by the flip*
endWhile
Flip the four tetrahedra incident to x^+

Similarly to the algorithm for insertion, the main iteration performs a series of flips of faces (triangles), but in this case the faces are those incident to x^+ (instead to opposite to the vertex), and faces are stored in a sorted queue (instead of an unsorted list). Notice that the queue contains all the faces incident to x^+, flippable or not. The iteration stops when there are only four (not l-flippable) facets in the queue. At this time, a $4 \rightarrow 1$ flip can be performed that finally removes x from the $\mathcal{H}(S^*)$.

The sorting criterion for facets is derived from the labelling of the vertices implied in the flip. Given Δ_2 a triangular face to be flipped, we define the index of Δ_2 to be

the maximum label of vertices implied in the flip if Δ_2 is l-flippable, $+\infty$ otherwise,

$$index(\Delta_2) = \begin{cases} maxlabel(\Delta_2) & \text{if } \Delta_2 \text{ is l-flippable} \\ & \text{seen from } x^+ \text{ ,} \\ +\infty & \text{otherwise} \end{cases}$$

Facets in the priority queue are sorted by their indices, from lowest to highest, therefore non-flippable facets will be placed at the end of the queue. Each time a flip is performed, new l-flippable facets incident to x^+ created by the flip (if any) are added to the queue. In addition, since the indices of facets surrounding the flip will have changed, these facets must be removed from the queue and inserted again in its right place. Remark that some of these facets may change from not flippable to flippable.

Notice that to delete x^+ from $\mathcal{H}(S^*)$ we only need to modify tetrahedra incident to this vertex. The algorithm, in effect, only flips facets incident to x^+. Let j be the label of x, that is $x = v_j$. From now on, we will refer to x as v_j. We will see that the way the deletion algorithm works is performing successive rotations of v_j and posterior vertices $(v_{j+1}, v_{j+2}, \dots)$ until a minimun label k_0 is found such that v_j^+ lies above $LCH(S_{k_0}^+)$, that is to say, v_j is redundant in $\mathcal{RT}(S_{k_0})$.

Lemma 3. *Before entering to the main iteration, the index of the first face in the queue is greather than j.*

Proof. Only facets incident to v_j^+ are queued, and facets Δ_2 incident to v_j^+ with $maxlabel(\Delta_2) = j$ are not l-flippable seen from v_j^+. □

Observation: This means that in practice faces with maximum label j can be directly put at the end of the queue with index $+\infty$, without having to test for its l-flippability.

Since we are working with a sorted queue, we can group the lifted flips corresponding to a facets with the same index. We have yet seen that the index of the first element in the queue is $> j$; later we will see that this index does not decrease. In fact, the lifted flips corresponding to faces with index k perform the rotation of v_j^+ and v_k^+ (except for the last index, which is a special case of a rotation that finally removes v_j^+ from $\mathcal{H}(S^*)$).

Lemma 4. *If the first index of the queue is greater than $j+1$, then the tetrahedra in $\mathcal{H}(S^*)$ with vertices in $S_{j+1}^+ - \{v_j^+\}$ give a tetrahedrization of this set of vertices that fulfills Property 1.*

Proof. Since there is no face with index $j+1$ in the queue, there are no l-flippable facets seen from v_j^+ with index $j+1$. This implies that the lower hull of the tetrahedra with vertices in $S_{j+1}^+ - \{v_j^+\}$ is convex. □

Lemma 5. *Let k be the index of the first element in the queue and suppose the algorithm has not performed a flip of a facet whose index is k. Suppose also that v_j^+ lies below $LCH(S_k^+ - \{x^+\})$. Then,*

(a) The algorithm will only add faces in the queue whose index is greater or equal to k.

(b) When the index of the first element in the queue is greater than k, the algorithm will have performed the rotation of v_j^+ and v_k^+.

Proof. Using Lemma 4 recursively we can guarantee that when the algorithm reaches the hypothesis state the set $S_{k-1}^+ - \{v_j^+\}$ fulfills Property 1, that is to say, $\mathcal{H}(S^*)$ is yet updated for vertices in S_{k-1}. Notice that the algorithm in this state works equivalently to the insertion algorithm, i.e. it performs lifted flips seen from v_j^+, taking into account that tetrahedra incident to v_k^+ with a face $\Delta \in LCH(S_{k-1}^+ - \{v_j^+\})$ not visible from v_j^+ are not removed. Consequently, after all l-flippable facets with index k had been flipped, v_k^+ is joined to triangles in $V(S_{k-1} - \{V_j\}, v_j, v_k)$, and (once again, using the same arguments of topological correctness of lifted flips) x^+ is joined to visible facets in $LCH(S_k^+ - \{v_j^+\})$. Notice also that facets with index less than k are not modified. Therefore, the rotation has been achieved.

Furthermore, since lifted flips seen from v_j^+ only create new tetrahedra incident to v_k^+, only faces can with label $\geq k$ will be enqueued or reordered in the queue. □

Theorem 2. *The deletion algorithm obtains $\mathcal{H}(S^* - \{v_j^+\})$ fulfilling Property 1.*

Proof. Let k_0 be the label of the vertex in S^* such that v_j^+ is comprised between $LCH(S_{k_0}^+ - \{v_j^+\})$ and $LCH(S_{k_0-1}^+ - \{v_j^+\})$. Then, since the algorithm only flips faces around v_j^+, the indices of the facets in the queue are all $\leq k_0$. Using Lemmas 3 and 5, with a finite number of steps we can reduce to case $k_0 = j+1$, that is $v_{k_0} = v_{j+1}$. Similarly to the case of the insertion, the lifted flips of faces whose index is $j+1$ can be understood as a special case of a rotation. In this case, the vertex v_j^+ is above $LCH(S_{j+1}^+ - \{v_j^+\})$. The flips performed of faces whose index is $j+1$ create tetraedra joining vertex v_{j+1}^+ with all visible faces in $V(S_{j-1}, v_j, v_{j+1})$, except for the face f of $LCH(S_{j-1}^+)$ such that $v_j^+ \in CH(f \cup v_{j+1}^+)$. The last $1 \to 4$ flip finally removes v_j^+ from $\mathcal{H}(S^*)$. □

7.1 A counterexample for unsorted flipping in deletion

The deletion algorithm presented in previous section uses a sorted queue of facets, therefore flips are performed in a particular order. Can the algorithm be modified so that the queue is not sorted? In other words: can the flips be done in any arbitrary order? In this section we show that the answer to this question is negative, with the help of a counterexample. Notice that unsorted flipping could not maintain a triangulation that fulfills Property 1.

More precisely, the unsorted flipping version of the algorithm to delete a vertex from a regular triangulation works for a set of weighted points in $\mathbb{R}^1$ (i.e., for a convex hull in $\mathbb{R}^2$), however in higher dimensions the algorithm may not converge. Consider the set of seven weighted points in $\mathbb{R}^2$ such its mapping in $\mathbb{R}^3$ is the one given in Figure 8, and suppose that the lowest point, point number 4, is the one to be deleted. Depending on the order the points were inserted in $\mathcal{RT}(S)$, the unsorted flipping can fall in a situation like the one represented in the figure, where none of the facets is flippable but the vertex is incident to more than four tetrahedra, that is, we have not

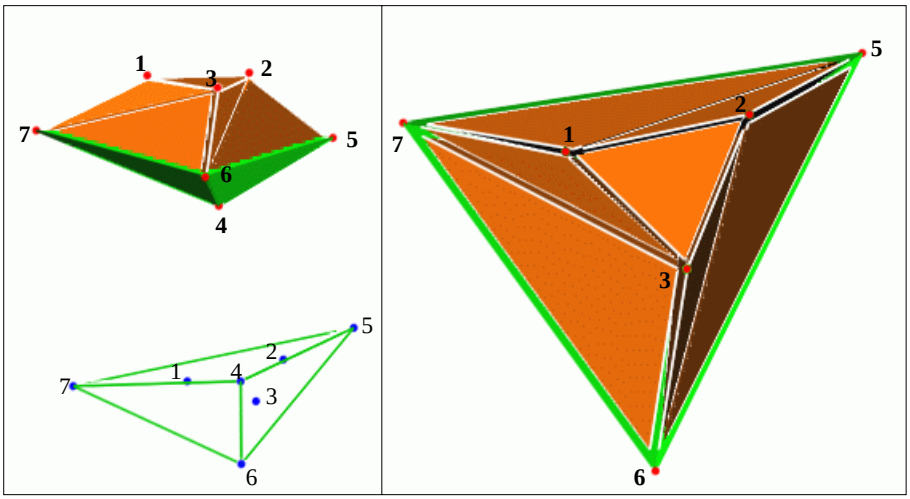

Figure 8: Two projective views of the counterexample set which shows that unsorted flipping edges for deletion may not work. Vertex number 4 is the one being deleted. *Left*, the set of points (in red), lower convex hull (in green), tetrahedra incident to fourth point (in orange), and the regular triangulation projected into the plane; *right*, a top view of the same example.

succeeded to delete it. (In the figure, only tetrahedra incident to the fourth vertex are represented for a better comprehension.) The configuration of this counterexample is the so-called Shönhardt polytope in $\mathbb{R}^3$ with an additional lower point (the observer vertex) connected to it. Notice that the unique possible tetrahedrization of this polyhedron is the one with edges connecting point number 4 to the other six points. Notice also the ressemblance between this counterexample and the one given in [8] to prove that arbitrarly edge flipping does not work for regular triangulations.

8 Results and conclusions

We have developed algorithms for managing regular triangulations of a dynamic sets of points, in which points can be inserted and deleted arbitrarily. The two algorithms are quite simple thanks to the fact that they only use an auxiliary data structure geometrically interpretable, the historical triangulation of the paraboloid projection in $\mathbb{R}^{d+1}$. Treatment for redundant and non-redundant points on the triangulation could have been unified using an auxiliary vertex with infinite weight, sv_4. The algorithms have been implemented, and their performance has been checked.

In order to evaluate the insertion algorithm, it has been tested on a set of 1000 points in $\mathbb{R}^2$ uniformly distributed in a circle of radius 200, and weights between 0 and 100. The resulting regular triangulation has 188 vertices (the remaining points are redundant) and 340 triangles. In Figure 9 the CPU time for incrementally inserting each point is represented. Data show a logarithmic expected time behavior of the algorithm.

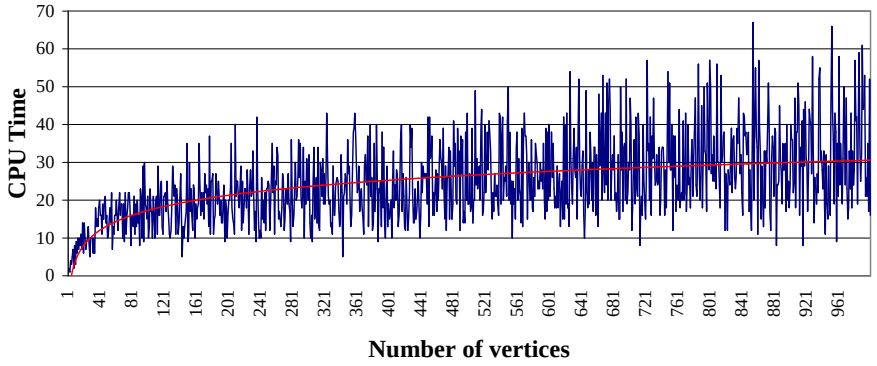

Figure 9: CPU time of insertion for computing the regular triangulation of 1000 points. Red curve is a logarithmic interpolation of data.

The total time for creating the regular triangulation of the whole set was 3.57 seconds on a Sun/Sparc 5.7 Solaris machine.

Using the same set of 1000 points, the deletion algorithm has been evaluated. 100 different points have been deleted from the original set, and each time the number of lifted flips has been counted. As it could be expected, the deletion cost depends on the label assigned to the point. Figure 10 shows the number of flips depending on the insertion order of each vertex being deleted. Remark that insertion order does not coincide with the label of a vertex, thus the deletion algorithm performs many flips when removing vertices that have been inserted lately but are assigned a small label (hence the peaks in the right part of the diagram). In fact, while the mean number of flips for deletion a vertex is 8.35, for vertices with insertion order less than 500 the mean is 14.48, and for the rest of vertices the mean is 2.22. Furthermore, while the mean deletion time of a single vertex is less than 1 millisecond, the maximum is 7 milliseconds.

Although the proposed algorithms have been designed for adding and removing points in regular triangulations, it is not difficult to modify them so that dynamic convex hulls in $\mathbb{R}^d$ are maintained. Indeed, it suffices to assign label $+\infty$ to all the super-vertices, so that they all were kept as the last vertices inserted in the set. Then, the convex hull of the set S^+ can be obtained simply removing the simplices incident to vertices in SV. Another way for obtaining $CH(S^+)$ from the history data structure consists on applying the deletion algorithm to each super-vertex (except for the last) and then removing the simplices incident to them. Remark that in this case the deletion algorithm cannot converge to a situation where the super-vertex is surrounded by only four tetrahedra, but since it reorders the insertion labels of vertices so that the deleted vertex was the last being inserted, removing the simplices incident to the super-vertex the convex hull of the remaining set is obtained. Notice also that usually regular triangulations are not defined for a set of points with two (or more) points with same

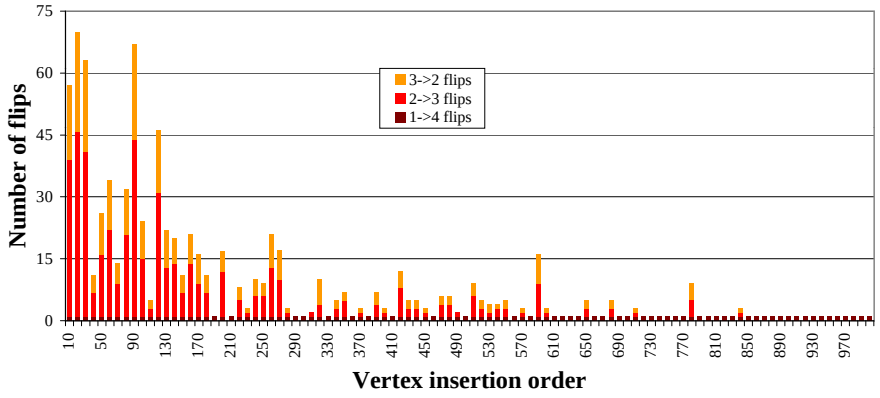

Figure 10: Number of flips performed by the deletion algorithm depending on the vertex label.

coordinates and different weights (in this case, one of the two points must be redundant). However, situations like that can be treated by the proposed algorithms without any modification.

Besides to this modification of the algorithm for managing dynamic convex hulls, algorithms for regular triangulations are also useful elsewhere Delaunay triangulations are used. Our proposal is specially appropriated wherever dynamic sets of points are required or weighting points can be of some utility. This could be the case of FEM analysis, since mesh refinement is often required for unstructured methods, and adaptive methods can take advantage of the extra degree of freedom supplied by the weight. Two recent applications are given in reference [3], where regular triangulations are used to improve the shape of a tetrahedrization, and in reference [9], which makes an intensive usage of the history data structure to obtain adaptive simplicial grids from its storage in one more dimension. A specific application where the authors currently are working in is mesh simplification for solid modeling, where weights indicate the probability of the vertices to be removed from the mesh. Other application fields include surface approximation, creation of levels-of-detail objects, computational fluid dynamics and robot movement planning.

9 Acknowledgments

Research of the authors has been partially supported by the CICYT under grants TIC-98-0586-603-01 and TIC-99-1230-102-02.

References

[1] F. Aurenhammer. Voronoi diagrams - A survey of a fundamental geometric data

structure. *ACM Computing Surveys*, 23(3):345–405, September 1991.

[2] J. Boissonat and M. Teillaud. On the randomized construction of the Delaunay tree. *Theoretical Computer Science*, 112:339–354, 1993.

[3] S.-W. Cheng, T. Dey, H. Edelsbrunner, M. Facello, and S.-H. Teng. Sliver exudation. In *15th ACM Sym. on Comp. Geometry*, pages 1–13, 1999.

[4] K. L. Clarkson, K. Mehlhorn, and R. Seidel. Four results on randomized incremental constructions. *Comput. Geom. Theory Appl.*, 3(4):185–212, 1993.

[5] O. Devillers. Improved incremental randomized delaunay triangulation. In *14th Annu. ACM Sympos. Comput. Geom.*, pages 106–115, 1998.

[6] O. Devillers. On deletion in Delaunay triangulations. In *Proc. 15th Annu. ACM Sympos. Comput Geom.*, pages 181–188, 1999.

[7] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag, Heidelberg, 1987.

[8] H. Edelsbrunner and N. R. Shah. Incremental topological flipping works for regular triangualtions. In *Proc. 8th Annual ACM Symposium on Computational Geometry*, pages 43–52, June 1992.

[9] H. Edelsbrunner and R. Waupotitsch. Adaptive simplicial grids from cross-sections of monotone complexes. *Int. Journal of Comput. Geom. and Appl.*, 10:267–284, 2000.

[10] M. Facello. Implementation of a randomized algorithm for Delaunay and regular triangulations in three dimensions. *Computer Aided Geometric Design*, 12(4):349–370, 1995.

[11] L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Transactions on Graphics*, 4(2):74–123, 1985.

[12] L. J. Guibas, D. E. Knuth, and M. Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, 7:381–413, 1992.

[13] C. Lawson. Software for C1 surface interpolation. In J. Rice, editor, *Mathematical Software III*, pages 161–194. Academic Press, 1977.

[14] T. Masada, H. Imai, and K. Imai. Enumeration of regular trtiangulations. In *Proc. ACM Computational Geometry*, pages 224–233. ACM, 1996.

[15] E. Mücke, I. Saias, and B. Zhu. Fast randomized point location without preprocessing in two- and three-dimensional Delaunay triangulations. *Computational Geometry*, 12:63–83, 1999.

[16] A. Okabe, B. Boots, and K. Sugihara. *Spatial Tessellations. Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons Ltd, 1992.

[17] F. Preparata and M. Shamos. *Computational Geometry*. Springer-Verlag, 1985.