Updating and Constructing Constrained Delaunay and Constrained Regular Triangulations by Flips

Jonathan Richard Shewchuk
Department of Electrical Engineering and Computer Sciences
University of California at Berkeley
Berkeley, California 94720
jrs@cs.berkeley.edu

Abstract

I discuss algorithms based on bistellar flips for inserting and deleting constraining $(d-1)$-facets in d-dimensional constrained Delaunay triangulations (CDTs) and weighted CDTs, also known as constrained regular triangulations. The facet insertion algorithm is likely to outperform other known algorithms on most inputs. The facet deletion algorithm is the first proposed for $d > 2$, short of recomputing the CDT from scratch. An incremental facet insertion algorithm that begins with an unconstrained Delaunay triangulation can construct the CDT of a ridge-protected piecewise linear complex with n_v vertices in $\mathcal{O}(n_v^{\lfloor d/2 \rfloor + 1} \log n_v)$ time. Hence, in odd dimensions, CDT construction by incremental facet insertion is within a factor of $\log n_v$ of worst-case optimal. Perhaps the most important feature of these algorithms is that they are relatively easy to implement.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

General Terms

Algorithms, Theory

Keywords

Constrained Delaunay triangulation, bistellar flip

1. Introduction

A *constrained Delaunay triangulation* (CDT) is a variation of a Delaunay triangulation that is constrained to respect the shape of a domain—perhaps an object to be rendered, or a domain to be simulated by a numerical method like the finite element method. CDTs have desirable properties that make them useful in interpolation and numerical analysis, including their tendency to favor "round" tetrahedra over "skinny" (high aspect ratio) tetrahedra, their suitability for interpolation [25], and their mathematical properties that allow Delaunay refinement algorithms [4, 22] to generate meshes that have provably good characteristics.

A *regular triangulation* is a triangulation that can arise as a side view of a convex polytope. Delaunay triangulations are a special case of regular triangulations. A *constrained regular triangulation* is a triangulation that arises as a side view of a polytope that is locally convex everywhere except at the faces that are constrained to be part of the triangulation. (This notion is formalized shortly.)

Cheng et al. [2] have shown that regular triangulations are useful in three-dimensional mesh generation. Constrained regular triangulations are even more useful because of their ability to respect the shape of a domain. Another use for constrained regular triangulations, as this paper shows, is that they help in reasoning about algorithms for updating CDTs based on elementary geometric operations known as bistellar flips.

This paper discusses flip-based algorithms for updating and constructing CDTs and constrained regular triangulations. The algorithms are relatively simple (as compared to sweep algorithms [23]), yet are fast in odd dimensions, and are probably the best existing choice for practical three-dimensional CDT construction. Flip-based CDT construction takes $\mathcal{O}(n_v^{\lfloor d/2 \rfloor + 1} \log n_v)$ time, where n_v is the number of vertices in the input and d is the dimension. This is within a factor of $\log n_v$ of worst-case optimal in odd dimensions.

A CDT is a triangulation of an underlying input called a *piecewise linear complex* (PLC), following Miller, Talmor, Teng, Walkington, and Wang [17]. A PLC X is a set of *facets* of dimensions 0 through d. The 0-facets are vertices, and every vertex of a CDT of X is a vertex in X. Each higher-dimensional facet is a polytope (roughly speaking), possibly with holes, slits, and isolated vertices in it, as Figure 1 shows. Formally, a k-facet is a union of open convex k-polytopes lying in a common k-flat, although sometimes it is more convenient to think of the closure of the k-facet. A facet may be nonconvex and may have any number of faces. A facet need not be connected.

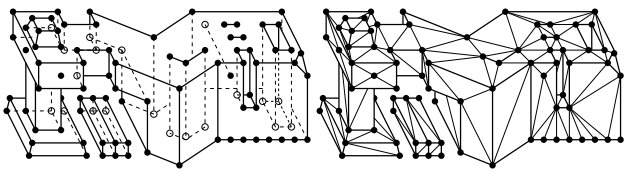

Figure 1: Each facet of a PLC (left) may have holes, slits, and interior vertices, which may be used to enforce the presence of specific faces (perhaps so that boundary conditions may be applied) or to support intersections with other facets. The right illustration is the constrained Delaunay tetrahedralization of the PLC.

Supported in part by the National Science Foundation under Awards ACI-9875170, CMS-9980063, CCR-0204377, and EIA-9802069, and by a gift from the Okawa Foundation.

SoCG'03, June 8–10, 2003, San Diego, California, USA.

A PLC must obey the same requirements as any type of complex: every lower-dimensional face of a facet in X is a facet in X, and every nonempty intersection of two facets in X is either a lower-dimensional facet in X or one of the two original facets.

The facets of dimension 1 through $d-1$ are called *constraining facets*, as they constrain the CDT: for a triangulation T to be a CDT of X, each constraining facet of X must be a union of faces of T. (A full definition of *CDT* appears in Section 2.)

The union of the d-facets of X is the *triangulation domain*, the region of space a user wishes to triangulate. X must contain the faces of the d-facets, so the triangulation domain is *facet-bounded*.

Updating a CDT is in some ways like updating a Delaunay triangulation, but there are catches. The first catch is that every modification is done in the context of an underlying PLC. It is not usually possible to determine how a CDT will change when a vertex or facet is inserted or deleted without knowing the PLC that determines it. Every incremental operation changes the PLC and the CDT together. The second catch is that the modified PLC might not have a CDT (or even a triangulation).

To *insert a constraining facet* into a CDT is to add the facet to the underlying PLC, and to update the CDT so it respects the new facet. To *delete a constraining facet* from a CDT is to remove the facet from the underlying PLC, and to update the CDT so it is no longer constrained by the facet (and it becomes "closer" to Delaunay).

Section 4 describes facet insertion and deletion algorithms that modify a CDT through a sequence of *bistellar flips*—topological transformations that locally replace one small set of d-simplices with another. (Flip-based vertex insertion and deletion are already more or less covered in the literature [10, 23]—see the end of Section 2.) These algorithms have the advantage of being especially easy to implement, for two reasons. First, they are substantially simpler than sweep algorithms. Second, the triangulation always maintains the geometric and topological relationships between different pieces of the mesh data structure. By contrast, incremental algorithms not based on flips typically require auxiliary data structures to keep track of dangling pieces while holes are being retriangulated.

These algorithms can be cast as a kinetic data structure [11] for maintaining the constrained regular triangulation of a set of vertices whose "heights" change linearly with time. Intuitively, it is easy to visualize the unconstrained case: a kinetic algorithm maintains the lower boundary of the convex hull of a set of vertices in E^{d+1} as they move parallel to the x_{d+1}-axis. When two (or more) facets of the convex hull become parallel, or a vertex recedes into the hull's interior, the hull is locally updated by a bistellar flip.

Kinetic maintenance of constrained regular triangulations is similar, but requires extra care to ensure that the dynamically changing PLC does not enter a configuration for which no constrained regular triangulation exists. The algorithms FLIPINSERTFACET and FLIPDELETEFACET, described in Section 4, schedule the vertex heights in a sneaky way so as to insert or delete a $(d-1)$-facet. If f is the facet being inserted or deleted, and X and X^f are the PLCs without and with f respectively, these algorithms work if both X and X^f have CDTs.

2. Weighted Constrained Delaunay Triangulations

Consider the Euclidean space E^{d+1}, and let $x_1, x_2, \ldots, x_{d+1}$ be the coordinate axes. E^d is the subspace of E^{d+1} orthogonal to the x_{d+1}-axis.

A d-dimensional triangulation is *regular* if and only if it is the vertical projection of one "side" of some $(d+1)$-dimensional convex polytope. Specifically, let P be any nondegenerate convex polytope in E^{d+1}. Say that a d-face f of P is *downward-facing* if a vector normal to f pointing away from P has a negative x_{d+1}-coordinate. Suppose that every downward-facing d-face of P is a simplex. Let T be the d-dimensional triangulation formed by vertically projecting the downward-facing faces of P onto E^d (by dropping the x_{d+1}-coordinate of each vertex). T is a regular triangulation.

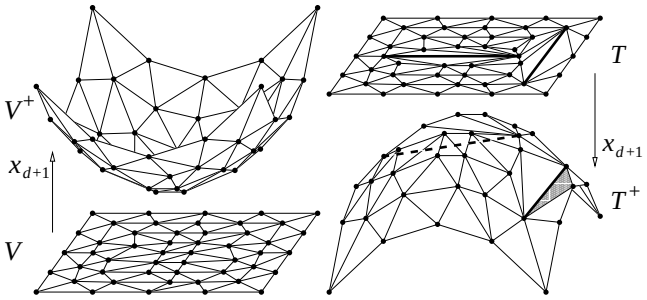

Figure 2: Left: The parabolic lifting map. In this example, a two-dimensional vertex set V is lifted to a paraboloid in E^3. A lifted Delaunay triangulation is the lower boundary of the convex hull of the lifted vertices. Right: A lifted CDT T^+. The paraboloid is inverted to more clearly show its topography. The bold edges are constraining edges that are not Delaunay. They are mapped to reflex edges of the lifted surface.

The best-known regular triangulation is the Delaunay triangulation. The regularity of Delaunay triangulations is demonstrated by the well-known *lifting map* of Edelsbrunner and Seidel [9]. Let V be a set of vertices in E^d for which a Delaunay triangulation is sought. The lifting map projects each vertex in V to a vertex on a paraboloid in a space one dimension higher, as Figure 2 (left) illustrates. Specifically, each vertex $v = (v_{x_1}, v_{x_2}, \ldots, v_{x_d}) \in V$ maps to a point $v^+ = (v_{x_1}, v_{x_2}, \ldots, v_{x_d}, v_{x_1}^2 + v_{x_2}^2 + \cdots + v_{x_d}^2)$ in E^{d+1}. Each pair of vertices v and v^+ are called *companions*, and v^+ is the *lifted companion* of v. Let V^+ be the set of lifted companions of the vertices of V. The Delaunay triangulation of V is regular because it has the same combinatorial structure as the lower convex hull of V^+. Each downward-facing d-face of the convex hull of V^+ maps to a Delaunay d-simplex of V. This connection is routinely used to transform any $(d+1)$-dimensional convex hull algorithm into a d-dimensional Delaunay triangulation algorithm.

A *weighted Delaunay triangulation* is similar to a Delaunay triangulation, but each vertex $v \in V$ is assigned a real-valued *weight* w_v. A vertex v is mapped to a point $v^+ = (v_{x_1}, v_{x_2}, \ldots, v_{x_d}, v_{x_1}^2 + v_{x_2}^2 + \cdots + v_{x_d}^2 - w_v)$. The weighted Delaunay triangulation of V is the projection of the downward-facing faces of the convex hull of the lifted points V^+. Every regular triangulation is a weighted Delaunay triangulation if the right weights are chosen, so I use the terms interchangeably.

Some faces of the convex hull of V^+ might not be simplicial, because some selection of $d+2$ or more of the lifted vertices may lie on a common non-vertical d-dimensional hyperplane. (Observe that vertices that lie on a common vertical hyperplane are of no concern, because a vertical face cannot be downward-facing. This is good news, because there may be many coplanar vertices in the input.) These non-simplicial faces can be triangulated in any compatible manner, so V has more than one weighted Delaunay triangulation. Section 5 describes a simple way of perturbing the weights to simulate the *nondegeneracy condition* wherein no $d+2$ vertices of V^+ lie on a common non-vertical hyperplane, thereby guaranteeing that the constructed triangulation is regular.

If its weight is sufficiently small, a vertex v^+ might not lie in the lower surface of the convex hull of V^+, in which case the vertex v is absent from the weighted Delaunay triangulation of V. Then v is said to be *submerged*. If every vertex has a weight of zero, the weighted Delaunay triangulation is the Delaunay triangulation, and no vertex is submerged, because every point of the paraboloid is in the lower convex hull of the paraboloid.

Simplices, like vertices, have companions. If s is a k-simplex with vertices $v_0, v_1, \ldots, v_k$, then its lifted companion s^+ is the k-simplex in E^{d+1} whose vertices are $v_0^+, v_1^+, \ldots, v_k^+$. Note that s^+ is flat; it does not curve to fit the paraboloid.

Let s be any simplex (embedded in E^d) whose vertices are in V. The simplex s is *regular* if s^+ is a face of the lower convex hull of V^+ (hence s appears in every weighted Delaunay triangulation of V). In other words, there exists a d-dimensional hyperplane h in E^{d+1} such that h includes s^+, and every vertex of V^+ not in s^+ lies above h. The hyperplane h is called a *witness* to the regularity of s. In weighted Delaunay triangulations, a witness serves the same purpose that a circumsphere serves in ordinary Delaunay triangulations.

The simplex s is *semiregular* if s^+ is included in a face of the lower convex hull of V^+. If s is semiregular but not regular, it lies in a nonsimplicial face of the convex hull, and appears in at least one weighted Delaunay triangulation of V. There exists a d-dimensional hyperplane h in E^{d+1} that is a *witness* to the semiregularity of s: h includes s^+, and every vertex of V^+ lies in or above h. If all the weights are zero, "semiregular" is equivalent to "Delaunay" and "regular" is equivalent to "strongly Delaunay."

What about *constrained* triangulations? Let X be a *weighted PLC*—a PLC whose vertices are assigned weights. There are applications where it is convenient for vertices with insufficient weight to be left out. Some vertices are allowed to be submerged (as their weights dictate), but some are not because they support constraining facets. A vertex of X is *submersible* if it does not lie on a segment in X, or if it is an endpoint of exactly two segments in X and those segments are collinear. In the latter case, think of the two segments as a single 1-facet of X. A row of collinear segments might form one 1-facet with many submersible vertices in it.

A simplex s *respects* a facet f if $s \cap \text{closure}(f)$ is a union of faces of s. This union may be the empty set, s itself, a lower-dimensional face, or the melding of several faces of s, possibly of mixed dimension. For example, a nonconvex facet might intersect two or even three edges of a triangle without including the triangle's interior. A simplex s *respects* X if it respects every constraining facet in X after agglomerating the segments of X into 1-facets as discussed above.

Say that the visibility between two points p and q in E^d is *occluded* if there is a constraining $(d-1)$-facet f in X such that p and q lie on opposite sides of the hyperplane that includes f, and the line segment pq intersects the closure of f. If either p or q lies in the hyperplane that includes f, then f does not occlude the visibility between them. Facets in X of dimension less than $d-1$ do not occlude visibility. The points p and q are *visible* from each other (equivalently, can *see* each other) if there is no occluding $(d-1)$-facet of X.

A simplex s whose vertices are in X is *constrained regular* if

- s respects X, and
- there exists a d-dimensional hyperplane h_s in E^{d+1} that includes s^+, such that every vertex v of X that is visible from any point in the relative interior of s, but is not a vertex of s, lifts to a point v^+ above h_s. (The hyperplane h_s is a witness to the constrained regularity of s.)

The second condition is a bit difficult to visualize, because one must simultaneously picture the vertices in E^d, where visibility is determined, and in E^{d+1}, where witness hyperplanes are defined. Think of it this way: if some lifted vertex v^+ lies below the hyperplane through a lifted simplex s^+, then s is not regular. But if some constraining $(d-1)$-facet of X occludes the view of v from inside s, s may still be constrained regular and appear in the triangulation. The shaded triangle in Figure 2 (right) is an example (note that the paraboloid in the figure is inverted for clarity). As the figure shows, each constraining $(d-1)$-simplex of the CDT that is not constrained regular is mapped to a reflex ridge in the lifted surface.

The term *constrained semiregular* means the same thing as constrained regular, except that any lifted vertex may lie in (but not below) h_s. Any constrained regular simplex is constrained semiregular, but the converse is not true. If all the weights are zero, "constrained semiregular" is equivalent to "constrained Delaunay."

A *weighted CDT* T of X is a triangulation that *fills* X (i.e. the union of the simplices in T is the union of the facets in X) wherein every simplex is constrained semiregular within the lowest-dimensional facet of X that includes it. Every simplex of T that does not lie in a constraining facet is therefore constrained semiregular in X, as defined above. However, if a simplex s in T is included in a k-facet f of X with $k < d$, then s is not required to be constrained semiregular in X. Rather, s is a simplex in the k-dimensional weighted CDT of f. The definition of "weighted CDT" is therefore recursive in the dimension.

In a weighted CDT, like in a weighted Delaunay triangulation, some of the vertices may be submerged. If all the weights are zero, every vertex is regular, so no vertex is submerged.

Throughout this paper, the terms "PLC" and "CDT" refer to both unweighted and weighted PLCs and CDTs, except where otherwise noted. A *hyperface* is a $(d-1)$-simplex and a *ridge* is a $(d-2)$-simplex.

One of the difficulties of working with CDTs is that not every PLC has one (except in the two-dimensional unweighted case). However, every *ridge-protected* PLC has a CDT. (See elsewhere [21] for a proof in the unweighted case; it is also true for weighted PLCs.) In two dimensions, a PLC X is ridge-protected if every non-submersible vertex in X is regular. All unweighted two-dimensional PLCs are ridge-protected; hence the success of CDTs in two dimensions. In three dimensions, X is ridge-protected if every constraining 1-facet in X is a union of regular edges. (Observe that this condition implies that every non-submersible vertex is regular.) In higher dimensions, X is ridge-protected if every constraining facet in X of dimension $d-2$ or less is a union of regular simplices.

A *constraining simplex of* X is a simplex in the CDT of a constraining facet in X. If a CDT of X exists, every constraining facet in X is therefore a union of constraining simplices of X. Note that a constraining simplex of X is not necessarily in X.

Ridge-protection implies that a weighted Delaunay triangulation (unconstrained) of the vertices of X contains all the constraining k-simplices of X for $k \leq d-2$, but might not respect the $(d-1)$-facets of X. Therefore, only the $(d-1)$-facets need to be inserted. If a PLC is not ridge-protected, it can be made ridge-protected with the addition of carefully chosen vertices with carefully chosen weights. If the PLC is unweighted and must remain unweighted, there is an algorithm for the three-dimensional case that makes a PLC ridge-protected by adding vertices that are placed so that the CDT of the augmented PLC does not have unreasonably short edges [24]. This algorithm is a good first step for guaranteed-quality mesh generation [22].

Because constraining facets of dimensions 1 through $d-2$ do not affect visibility, their insertion or deletion does not change the CDT of a PLC, except perhaps to change whether or not a CDT exists.

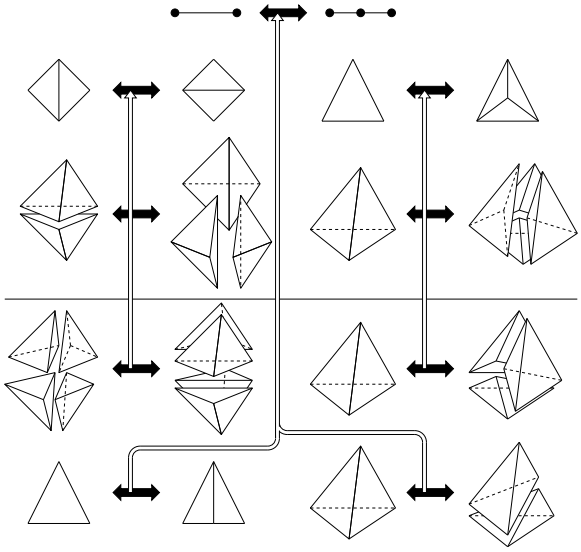

Figure 3: Basic bistellar flips in one, two, and three dimensions appear above the line. "Degenerate" bistellar flips appear below the line. White arrows connect degenerate flips to the lower-dimensional flips they are based on.

Therefore, general CDT maintenance can be done with four incremental operations, namely the insertion and deletion of vertices and $(d-1)$-facets. The vertex insertion algorithm of Edelsbrunner and Shah [10] for regular triangulations is easily adapted to weighted CDTs, simply by modifying the algorithm so it never performs a flip that breaks through a constraining facet. Vertex insertion takes $\mathcal{O}(m_{si})$ time, where m_{si} is the number of d-simplices deleted by the operation. The simple sweep algorithm for vertex deletion in CDTs [23] can be adapted so that it effects all structural changes through a sequence of flips. Vertex deletion takes $\mathcal{O}(m_{sf} \log m_{sf})$ time, where m_{sf} is the number of d-simplices constructed by the operation. This leaves the insertion and deletion of $(d-1)$-facets, which are covered in Section 4.

3. Bistellar Flips

Bistellar flips can be classified as nondegenerate and degenerate flips. A nondegenerate bistellar flip in E^d retriangulates the convex hull of $d+2$ vertices by replacing a collection of k d-simplices with $d+2-k$ different d-simplices. For instance, in three dimensions, a bistellar flip can replace two tetrahedra with three, three with two, one with four, or four with one. The one-to-four and four-to-one flips have the effect of inserting or deleting a vertex, respectively.

A degenerate bistellar flip is a flip that would be nondegenerate in a lower dimensionality, applied to lower-dimensional faces of a triangulation. A three-dimensional example is the insertion of a vertex into an edge. The vertex insertion replaces the edge with two edges, and is in essence a one-dimensional flip. However, every tetrahedron that shares the edge is divided into two tetrahedra, and the number of tetrahedra that share the edge could be arbitrarily large. Therefore, a degenerate flip can remove and create many d-simplices. A catalogue of bistellar flips for $1 \leq d \leq 3$ appears in Figure 3.

The flip algorithms described in Section 4 use flips to maintain the constrained regular simplices of a PLC with dynamically changing vertex weights. Say that a hyperface f, sandwiched between two d-simplices s and t in a triangulation, is *locally regular* if the lifted d-simplices s^+ and t^+ adjoin each other at a dihedral

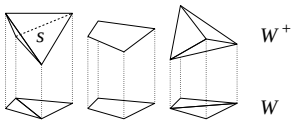

Figure 4: The bottom faces of a tetrahedron triangulate four vertices in the plane. Reversing the orientation of the tetrahedron (by moving its vertices vertically) switches to the other triangulation.

angle, measured from above, of less than 180°. In other words, the apex of t^+ lies above the witness hyperplane of s, and vice versa. For convenience, say that every hyperface included in only one d-simplex is locally regular as well.

It follows from a generalization of the Delaunay Lemma [7] that a triangulation that fills X, respects X, and has no submerged vertices is a weighted CDT if and only if every hyperface is a constraining simplex or is locally semiregular. Each flip algorithm waits until a non-constraining hyperface of the triangulation is no longer locally regular, then performs a flip that restores the local regularity of the non-constraining hyperfaces and therefore produces an updated CDT.

Let W be a minimal affinely dependent subset of vertices in E^d. *Minimal affinely dependent* means that although the vertices in W are affinely dependent, if any vertex is removed from W, the remaining vertices are affinely independent. Let $j+2$ be the number of vertices in W; j cannot exceed d.

One way to characterize a bistellar flip, useful for understanding regular triangulations, is to look at the set of lifted companions W^+ of the vertices in W. Suppose the vertex weights are chosen so that the vertices in W^+ are affinely independent. Then the convex hull of W^+ is a $(j+1)$-simplex s, as Figure 4 illustrates. Two possible triangulations of W are seen by looking at s from directly below, and from directly above. The downward-facing faces of s, projected down to E^d, form the regular triangulation of W. The upward-facing faces of s are not regular.

Suppose the vertices in W^+ are moving linearly. At some point in time, s may become flattened (Figure 4, center). At this instant, the nondegeneracy condition is not satisfied: the vertices in W^+ lie on a common non-vertical hyperplane. If the vertices continue moving, the downward-facing faces and the upward-facing faces of s trade places, and the triangulation of W that was formerly not regular becomes regular.

Lawson [15] shows that these are the only two ways to triangulate W. The two triangulations are characterized by Radon's Theorem [18], which states that there is exactly one way to partition W into two disjoint sets W_1 and W_2 so that $\mathrm{conv}(W_1)$ intersects $\mathrm{conv}(W_2)$. Figure 5 shows examples of these disjoint sets and the *Radon points* $\mathrm{conv}(W_1) \cap \mathrm{conv}(W_2)$. One triangulation of W consists of the j-simplices $T_1 = \{\mathrm{conv}(W - \{w\}) : w \in W_2\}$, and all their faces. Observe that all these j-simplices share the face $\mathrm{conv}(W_1)$. The other triangulation of W is composed of $T_2 = \{\mathrm{conv}(W - \{w\}) : w \in W_1\}$ and their faces. These j-simplices share the face $\mathrm{conv}(W_2)$.

A bistellar flip replaces T_1 with T_2 or vice versa. At least one of W_1 and W_2 has no more than $\lfloor j/2 \rfloor + 1$ vertices. Every bistellar flip either removes or creates a simplex of dimension $\lfloor j/2 \rfloor$ or less, namely $\mathrm{conv}(W_1)$ or $\mathrm{conv}(W_2)$. A degenerate flip either removes or creates at least one simplex of dimension $\lfloor d/2 \rfloor$ or less for every $2d+2$ d-simplices it removes or creates. These facts are handy for bounding the total number of flips that can occur.

If $j < d$, the flip is degenerate. In a d-dimensional triangulation T, one cannot simply replace T_1 with T_2 in isolation, because the

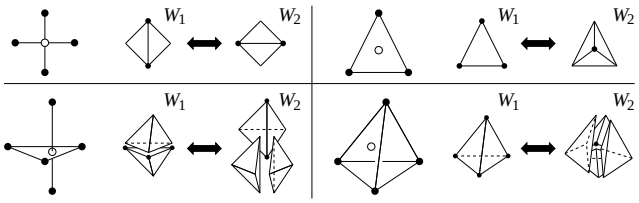

Figure 5: Four examples of Radon points (white circles) and the flips related to them.

FLIP(T, g)
{ T is a triangulation. g is a hyperface to flip. This procedure assumes g can be flipped. Comments below identify places where this assumption could fail. }
  Let s and t be the d-simplices that include g
  $W_R \Leftarrow \emptyset$
  $W_C \Leftarrow$ { the vertices of s and t not shared by g}
  $j \Leftarrow d$
  for each ridge (i.e. $(d-2)$-simplex) r of g
    $v \Leftarrow$ the vertex of g not shared by r
    $\theta \Leftarrow$ the exterior dihedral angle of $s \cup t$ at r
    if $\theta > 180^\circ$
      $W_R \Leftarrow W_R \cup \{v\}$
    else if $\theta < 180^\circ$
      $W_C \Leftarrow W_C \cup \{v\}$
    else $j \Leftarrow j - 1$ { v not in minimal affinely dep. subset }
  $T_R \Leftarrow \{\text{conv}(W_R \cup W_C - \{v\}) : v \in W_C\}$
  $T_C \Leftarrow \{\text{conv}(W_R \cup W_C - \{v\}) : v \in W_R\}$
  if $j = d$
    { note: if the simplices in T_R are absent from T,
        the flip cannot be performed }
    Delete each d-simplex in T_R from T
    Add each d-simplex in T_C to T
  else
    { degenerate flip; the members of T_R, T_C are j-simplices }
    Let y be any simplex in T_R
    for each $(d-j-1)$-simplex z such that $\text{conv}(y \cup z)$ is
        a d-simplex of T
      { note: if the simplices deleted below are absent from T,
          the flip cannot be performed }
      for each j-simplex y' in T_R
        Delete the d-simplex $\text{conv}(y' \cup z)$ from T
      for each j-simplex y' in T_C
        Add the d-simplex $\text{conv}(y' \cup z)$ to T

Figure 6: Algorithm for performing a bistellar flip (except the flips that insert a new vertex). An implementation should not calculate θ; rather, the tests "$\theta <> 180^\circ$" should branch based on the results of a vertex orientation test.

members of T_1 are faces of d-simplices in T which must be flipped as well, as the degenerate examples in Figure 3 illustrate. The details are embodied in the last six lines of the pseudocode for the FLIP procedure in Figure 6.

FLIP determines what type of bistellar flip to perform to remove a hyperface g and the two d-simplices s and t that share it. The procedure first determines the vertex sets W_R, the vertices of the common face removed by the flip, and W_C, the vertices of the common face created by the flip. Let $W = W_R \cup W_C$. FLIP removes the simplices $T_R = \{\text{conv}(W - \{v\}) : v \in W_C\}$, and creates the simplices $T_C = \{\text{conv}(W - \{v\}) : v \in W_R\}$. If g was locally regular before the flip occurred, then the simplices in T_R (plus their lower-dimensional faces) formed the weighted Delaunay triangulation of W. FLIP is called at the moment when g is no longer locally regular. At that moment, the shared face $\text{conv}(W_R)$ is no longer constrained regular, so neither $\text{conv}(W_R)$ nor any simplex that includes it can remain in the CDT. After the flip, the simplices in T_C, plus their lower-dimensional faces, are the weighted Delaunay triangulation of W instead. T_R and T_C are j-dimensional, so if $j < d$, FLIP removes every d-simplex that has a face in T_R, and creates new d-simplices that each have a face in T_C.

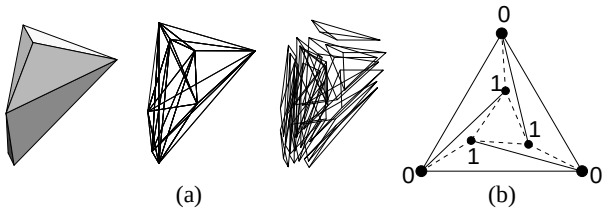

Figure 7: Stuck triangulations. (a) Several views of Joe's example of seventeen tetrahedra for which no triangular face that is not locally Delaunay can be flipped. (b) Edelsbrunner and Shah's example of seven triangles for which no edge that is not locally regular (dashed edges) can be flipped. Imagine that you are viewing the lifted triangulation from directly underneath, and larger nodes are closer to you. The number next to each vertex is the x_{d+1}-coordinate to which it is lifted (i.e. its distance from you). The regular triangulation of these vertices is the outer triangle; the inner vertices should be submerged.

There are several circumstances annotated in the code in which it might be impossible to perform a flip that eliminates g, because the initial triangulation is missing some of the simplices needed for a flip to take place. CDTs introduce more such cases, because flips are not allowed to penetrate constraining facets.

The main obstacle to designing a flip algorithm is that the algorithm might get stuck if it cannot perform a flip that makes progress toward a CDT. This is true even for unconstrained regular triangulations. Figure 7(a) depicts Joe's example [12] of a tetrahedralization whose locally non-Delaunay faces cannot be flipped. Figure 7(b) is a two-dimensional example of Edelsbrunner and Shah [10], wherein a flip algorithm gets stuck while trying to produce a weighted Delaunay triangulation. (For the special case of two-dimensional unweighted PLCs, flipping locally non-Delaunay edges always yields a CDT eventually [14, 16].)

Joe [13] shows that, if just one vertex (at a time) is inserted into a Delaunay triangulation of any dimension, flips can restore the regularity of the triangulation without needing a priority queue to avoid getting stuck. Edelsbrunner and Shah [10] show that the same is true for regular triangulations. Unfortunately, these results do not extend to facet insertion or deletion. The flip algorithms in the next section avoid getting stuck by using a priority queue to control the order in which flips occur, and scheduling the vertex weights carefully to ensure that constraining facets are respected.

For these algorithms, a critical aspect of scheduling is that the vertex weights must be perturbed. The purpose of the weight perturbations is to ensure that the retriangulation performed at any instant in time is never more complicated than a bistellar flip. Section 5 describes a symbolic perturbation method that ensures that the flip algorithms are correct. Each update algorithm requires that the input triangulation is a weighted CDT consistent with the perturbation method.

For more discussion of bistellar flips, see Lawson [15], Edelsbrunner and Shah [10], and de Loera, Santos, and Urrutia [6].

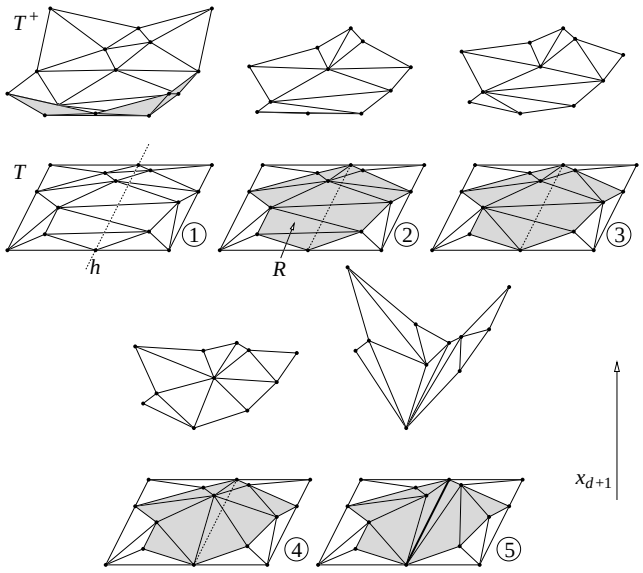

Figure 8: A two-dimensional example of inserting a facet into a CDT. The algorithm works in any dimensionality.

4. Flip-Based Facet Insertion and Deletion

This section describes algorithms, based on bistellar flips, for inserting or deleting a $(d-1)$-facet. These algorithms work correctly if both the initial and final PLCs (with and without the facet) have CDTs.

Consider inserting a $(d-1)$-facet f into a PLC X. Let $X^f = X \cup \{f\}$. The insertion of f is only possible if X^f is a valid PLC, which implies that all the lower-dimensional faces of f are present in X. Let T be the CDT of X. If X does not satisfy the nondegeneracy condition—that is, if $d+2$ vertices of X lift to a common non-vertical hyperplane—then T is required to be the CDT of X after X is perturbed as described in Section 5.

If every vertex of X appears in T, the algorithm FLIPINSERTFACET takes T as input and produces the CDT T^f of X^f (if a CDT exists). If vertices of X are submerged in T because their weights are too small, the insertion of f might make some of them be no longer submerged, but they do not appear in the triangulation FLIPINSERTFACET produces. To produce the CDT of X^f, either the triangulation must be postprocessed to incrementally insert the missing vertices, or the missing vertices must be removed from X^f. (This is not a defect in FLIPINSERTFACET; rather, inserting the missing vertices at the end is the fastest apparent way to compute T^f. Of course, most applications want unweighted CDTs, which do not have submerged vertices.)

A key observation is that inserting a facet does not create any new visibilities (though it might eliminate some old ones), so every simplex of T that respects f remains constrained regular, and therefore remains in T^f. Let R be the union of d-simplices of T that do not respect f. The flip algorithm only performs flips in the region R.

Let h be the hyperplane in E^d that includes f. Call the vertices of X on one (arbitrary) side of h *left vertices*, and the vertices on the other side *right vertices*. Vertices in h are neither. The flip algorithm linearly decreases the weights of the vertices according to their distance from h, and uses flips to maintain a locally regular triangulation of R as the weights change. Figure 8 is a sequence of snapshots of the algorithm at work.

For each vertex v in X, let its *height* k_v be the x_{d+1}-coordinate to which v is lifted—namely $\|v\|^2 - w_v$, where w_v is the weight of v. Each vertex v is assigned a time-varying height of $k_v(\tau) = k_v + \tau d(v,h)$, where τ is the time and $d(v,h)$ is the Euclidean distance of v from h. (This choice of $d(\cdot,\cdot)$ is pedagogically useful but numerically poor; a better choice for implementation is to let $d(v,h)$ be the distance of v from h along one coordinate axis, preferably the axis most nearly perpendicular to h. This distance is directly proportional to the Euclidean distance, but can be computed without radicals.)

When a set of vertices is transformed affinely, its convex hull undergoes no combinatorial change. Likewise, an affine transformation of the lifted vertices that changes only their x_{d+1}-coordinates does not topologically change the CDT. In the flip algorithm, however, each half of space undergoes a different affine transformation, so the CDT changes near the hyperplane h as τ increases. Observe that an algorithm in which only the heights of the right vertices change (at twice the speed) is equivalent, as is an algorithm in which only the left vertex heights change. For numerical reasons, it is better to raise only half the vertices.

Let $X(\tau)$ be a time-varying weighted PLC, which is identical to X except that each right vertex v has a height of $k_v(\tau)$. Any simplex s of T that has no right vertex remains constrained regular in $X(\tau)$ for all $\tau \geq 0$, because the witness hyperplane of s is constant and no vertex height decreases. Because an algorithm that raises only the left vertices is equivalent, any simplex of T that has no left vertex also remains constrained regular in $X(\tau)$ for all $\tau \geq 0$.

A simplex with no left or no right vertex remains always constrained regular. Recall that if a flip is initiated by a non-constraining hyperface losing its local regularity, the shared face $\mathrm{conv}(W_R)$ removed by the flip is not constrained regular. Thus, $\mathrm{conv}(W_R)$ has both a left vertex and a right vertex, and thus so does every simplex removed by the flip. The algorithm FLIPINSERTFACET removes only simplices that have a left and a right vertex *and* pass through f; all simplices outside the region R remain intact. Interestingly, the algorithm will work even if $X(\tau)$ does not have a CDT for some values of $\tau > 0$; the region R is always covered by simplices that are constrained regular in $X(\tau)$.

More good news is that no simplex of T that lies in the boundary of R has both a left vertex and a right vertex, because every simplex in T respects f's lower-dimensional faces, which are present in X. Therefore, the flip algorithm can restrict its attention to the region R without fear that a flip will try to break through the boundary of R.

As $\tau \to \infty$, the flip algorithm reaches a state where f is a union of faces of the triangulation, and no further flips can occur. At this time, the triangulation maintained by FLIPINSERTFACET is the CDT of X^f, and the job is done.

Pseudocode for FLIPINSERTFACET appears in Figure 9. The **while** loop dynamically maintains the triangulation T as τ increases from 0 to ∞ and the lifted companions of the right vertices move up. For certain values of τ, the following event occurs: some hyperface g in the region R is no longer locally regular at time τ, because the two d-simplices that include g have the same witness hyperplane. Upon this event, an update operation replaces these and other simplices that are no longer constrained regular with simplices that are constrained regular immediately after time τ. Because of the perturbation method, a single bistellar flip suffices to perform each update.

To ensure that it performs each bistellar flip at the right time, the algorithm maintains a priority queue that stores any flip that might occur. For each hyperface g that will be flipped at some time in the future, the procedure CERTIFY determines when g might be flipped and enqueues a flip event. The **while** loop repeatedly removes the

FLIPINSERTFACET(X, T, f)
{ X is a PLC. T is its CDT. f is the facet to insert. }
  Identify a d-simplex in T that intersects the relative interior of f by a rotary search around a ridge of f
  Identify every d-simplex in T that intersects the relative interior of f by depth-first search
  for each hyperface g of T that intersects the relative interior of f and has at least one vertex on each side of f
    CERTIFY(g)
  while priority queue Q is not empty
    Remove $\langle g', \tau \rangle$ with minimum τ from Q
    if g' is still a hyperface of T
      FLIP(T, g')
      for each hyperface g that lies in the boundary of the retriangulated region and has at least one vertex on each side of f
        CERTIFY(g)
  return T

CERTIFY(g)
  Let s and t be the d-simplices that have g for a face
  $\tau \Leftarrow$ the time at which $h_s = h_t$ (s^+ and t^+ are cohyperplanar)
  Insert $\langle g, \tau \rangle$ into priority queue Q

Figure 9: Algorithm for inserting a $(d-1)$-facet into a CDT. Works if T is the CDT of the initial PLC X, and the final PLC $X^f = X \cup \{f\}$ is a valid complex that has a CDT. CERTIFY assumes that each right vertex v has height $k_v(\tau) = k_v + \tau d(v, h)$.

flip with the least time from the priority queue, and performs a flip if the hyperface still exists (and was not eliminated by other flips). When the queue is empty, the algorithm returns the triangulation T^f.

The algorithm FLIPDELETEFACET, whose pseudocode appears in Figure 10, takes T^f as input and produces T, essentially by reversing the insertion algorithm. The algorithm is guaranteed to succeed if X and X^f both have CDTs. (Deleting a facet does not cause formerly submerged vertices to emerge, so FLIPDELETEFACET works correctly even if X and X^f have submerged vertices.) There are a few things that make the deletion algorithm a bit different from the insertion algorithm. First, the shape of the region R is unknown at the start; it is only apparent when the algorithm halts. Second, the algorithm begins with $\tau = -\infty$ and ends with $\tau = 0$, and each right vertex v is assigned a height $k_v(\tau) = k_v - \tau d(v, h)$.

The correctness proof for FLIPINSERTFACET is omitted, but here are a few highlights. By induction, the sequence of flips that FLIPINSERTFACET performs maintains the local semiregularity of all hyperfaces inside the region R. By the Delaunay Lemma, after each flip, the simplices in R form the CDT of the PLC defined by restricting $X(\tau)$ to R and adding the boundary of R.

When the flips are done, the original vertex weights are restored. This is an affine transformation of the lifted right vertices, so the hyperfaces inside R remain locally semiregular, except the hyperfaces in f (which are now constraining hyperfaces).

The non-constraining hyperfaces outside R and on the boundary of R are locally semiregular because they respect f and they were locally semiregular before f was inserted. By a final application of the Delaunay Lemma, the whole triangulation is a CDT of X^f.

For an analysis of the running time, let m_v be the number of vertices in the region R, and let m_{left} and m_{right} be the numbers of left and right vertices in R (which are counted in m_v as well).

FLIPDELETEFACET(X^f, T, f)
{ X^f is a PLC. T is its CDT. f is the facet to delete. }
  for each hyperface g of T included in f
    CERTIFY(g)
  while priority queue Q is not empty
    Remove $\langle g', \tau \rangle$ with minimum τ from Q
    if g' is still a hyperface of T
      FLIP(T, g')
      for each hyperface g in the boundary of the region retriangulated by the flip
        if g is not a constraining hyperface of $X^f - \{f\}$
          CERTIFY(g)
  return T

CERTIFY(g)
  Let s and t be the d-simplices that have g for a face
  $v \Leftarrow$ the vertex of t that is not a vertex of g
  if v^+ will lie below the witness hyperplane h_s of s at time $\tau = 0$
    $\tau \Leftarrow$ the time at which $h_s = h_t$ (i.e. v^+ lies in h_s)
    Insert $\langle g, \tau \rangle$ into priority queue Q

Figure 10: Algorithm for deleting a facet from a CDT. Works if T is the CDT of the initial PLC X^f, and the final PLC $X^f - \{f\}$ also has a CDT. CERTIFY assumes that each right vertex v has height $k_v(\tau) = k_v - \tau d(v, h)$.

Let m_{si} and m_{sf} be the number of d-simplices in R before and after a call to FLIPINSERTFACET or FLIPDELETEFACET. Let n_v be the number of vertices in T.

THEOREM 1. *The worst-case running time of* FLIPINSERTFACET *is in* $\mathcal{O}((m_{sf} + m_{\text{left}} m_{\text{right}} m_v^{\lfloor d/2 \rfloor - 1}) \log m_v + n_v)$.

PROOF. Every bistellar flip either removes or creates a simplex of dimension $\lfloor d/2 \rfloor$ or less; furthermore, it removes or creates at least one for every $2d + 2$ d-simplices it removes or creates. Because the vertex weights vary linearly with time, a simplex that loses the constrained regular property will never regain it; so once a simplex is removed, it is never created again. (The use of these observations to derive upper bounds is inspired by Seidel [19].)

Every $\lfloor d/2 \rfloor$-simplex that FLIPINSERTFACET removes has at least one left vertex and at least one right vertex. The algorithm removes fewer than $m_{\text{left}} m_{\text{right}} m_v^{\lfloor d/2 \rfloor - 1}$ such $\lfloor d/2 \rfloor$-simplices from the region R over its lifetime (including those it both creates and removes during execution).

The final triangulation of R has m_{sf} d-simplices, so the number of $\lfloor d/2 \rfloor$-simplices that FLIPINSERTFACET creates over its lifetime does not exceed $\mathcal{O}(m_{sf})$ plus the number it removes over its lifetime.

Therefore, FLIPINSERTFACET removes and creates at most $\mathcal{O}(m_{sf} + m_{\text{left}} m_{\text{right}} m_v^{\lfloor d/2 \rfloor - 1})$ d-simplices during its lifetime. The time spent performing flips is linear in this number. Each flip enqueues at most a constant number of events per d-simplex created. Each event costs $\mathcal{O}(\log m_v)$ time to enqueue and dequeue.

The cost of Line 1 (identifying a simplex that intersects the relative interior of f) is in $\mathcal{O}(n_v)$. This is a pessimistic running time; the worst case is achieved only if the initial ridge used for the search is a ridge of $\Theta(n_v)$ hyperfaces. All other costs are no greater than the costs above. ■

Miraculously, the worst-case time for any sequence of FLIPINSERTFACET operations is no greater than the worst-case time for one operation for which $m_v \sim n_v$.

THEOREM 2. *The worst-case running time of any sequence of valid* FLIPINSERTFACET *operations applied consecutively to a triangulation is in* $\mathcal{O}(n_v^{\lfloor d/2 \rfloor + 1} \log n_v)$ *(if Line 1 of* FLIPINSERTFACET *is implemented efficiently).*

PROOF. A sequence of FLIPINSERTFACET operations, like a single one, has the property that any simplex removed is never created again. Therefore, the sequence of operations creates fewer than $n_v^{\lfloor d/2 \rfloor + 1}$ $\lfloor d/2 \rfloor$-simplices, including those that are removed before the sequence ends, and removes fewer than $n_v^{\lfloor d/2 \rfloor + 1}$. The number of events enqueued is proportional to the number of d-simplices created, which is proportional to the number of $\lfloor d/2 \rfloor$-simplices created and removed. Each event costs $\mathcal{O}(\log n_v)$ time to enqueue and dequeue.

If the number of FLIPINSERTFACET operations exceeds $\Omega(n_v^{\lfloor d/2 \rfloor} \log n_v)$ (which is possible when d is odd but unlikely in practice), the cost of Line 1 of FLIPINSERTFACET must be reduced. This can be done by giving each constraining ridge of the triangulation a balanced search tree listing the adjoining hyperfaces in rotary order around the ridge. Thus, Line 1 executes in $\mathcal{O}(\log n_v)$ time. The balanced trees are updated in $\mathcal{O}(\log n_v)$ time per hyperface created or removed.

Line 2 (the depth-first search step) deserves attention because a d-simplex can intersect the interior of a facet without being removed by the facet's insertion, if only the boundary of the d-simplex intersects the facet. All the executions of Line 2 together take $\mathcal{O}(n_v^{\lfloor d/2 \rfloor + 1})$ time, because the total number of d-simplices that exist sometime during the sequence is in $\mathcal{O}(n_v^{\lfloor d/2 \rfloor + 1})$, and each d-simplex intersects the relative interior of at most a constant number of $(d-1)$-facets during its lifetime (not counting facets that are inserted after the d-simplex is removed). All other costs are no greater than the costs above. ■

An $\mathcal{O}(n_v^{\lfloor d/2 \rfloor + 1} \log n_v)$-time incremental algorithm for constructing a CDT of a ridge-protected PLC X follows from Theorem 2. First, construct the weighted Delaunay triangulation of the vertices in X in $\mathcal{O}(n_v^{\lceil d/2 \rceil})$ time using an incremental convex hull construction algorithm [5, 10]. Then insert the $(d-1)$-facets one by one.

How do flip algorithms compare to other algorithms in the literature? For unweighted two-dimensional PLCs, the fastest ways to construct a CDT, or to incrementally insert a segment, are the $\mathcal{O}(m_v \log m_v)$-time algorithms of Chew [3] and Seidel [20]. For weighted two-dimensional PLCs, Aichholzer, Aurenhammer, and Krasser [1] offer a flip algorithm that uses pseudo-triangulations to construct weighted CDTs. Its running time is unclear. For dimensions above two, the only alternative to the flip algorithms is a sweep-based algorithm called SWEEPCDT [23] (barring an unbearably slow gift-wrapping algorithm).

For constructing a CDT from scratch, SWEEPCDT takes $\mathcal{O}(n_v n_s)$ time, where n_s is the number of d-simplices in the CDT. Compare this with the $\mathcal{O}(n_v^{\lfloor d/2 \rfloor + 1} \log n_v)$ running time of flip-based incremental CDT construction. In the worst case, SWEEPCDT is faster in even dimensions, where $n_s \in \Theta(n_v^{d/2})$, and flip-based CDT construction is faster in odd dimensions, where $n_s \in \Theta(n_v^{\lceil d/2 \rceil})$. Recall that the greatest practical motivation for the flip algorithms comes from three-dimensional applications.

Often n_s is linear in n_v, and both algorithms may run faster. Recall from Section 3 that a bistellar flip replaces the bottom faces of a $(d+1)$-simplex with its top faces. Because no face reappears after it is removed, the sequence of flips performed during incremental facet insertion is structurally similar to a $(d+1)$-dimensional triangulation. It has often been observed that many practical vertex sets have linear-size Delaunay triangulations, and it seems likely that many practical inputs for the incremental facet insertion algorithm that have linear-size CDTs also engender only a linear number of flips. In the best case, for any $d \geq 2$, incremental CDT construction with FLIPINSERTFACET performs $\Theta(n_v)$ flips and runs in $\Theta(n_v \log n_v)$ time. The optimistic running time of SWEEPCDT is $\mathcal{O}(n_v n_f + n_v^{1+1/d})$, where n_f is the number of constraining $(d-1)$-simplices of X. Commonly $n_f \in \Theta(n_v)$, though n_f may range from $\Omega(1)$ to $\mathcal{O}(n_s)$.

There are inputs for which flip-based incremental CDT construction is asymptotically slower than SWEEPCDT, but these are surely the minority of inputs in practice when d is odd. For even d, the two algorithms may be more evenly matched.

Similar conclusions hold when FLIPINSERTFACET or SWEEPCDT is used to insert a single facet. In the best case, FLIPINSERTFACET takes $\Theta(m_v \log m_v)$ time, whereas removing the old simplices from R and retriangulating R using SWEEPCDT takes $\Theta(m_v^2)$ time. In the worst case in two dimensions, FLIPINSERTFACET takes $\Theta(m_v^2 \log m_v + n_v)$ time, and SWEEPCDT takes $\Theta(m_v^2 + n_v)$ time. In the worst case in three dimensions, FLIPINSERTFACET takes $\Theta(m_v^2 \log m_v + n_v)$ time, and SWEEPCDT takes $\Theta(m_v^3 + n_v)$ time.

A disadvantage of incremental CDT construction is that a facet cannot be inserted if its lower-dimensional faces are not already present in the CDT. SWEEPCDT can compute the CDT of almost any PLC that has a CDT (excepting some PLCs that fail to satisfy the nondegeneracy condition), whereas incremental CDT construction cannot, although it works for all ridge-protected PLCs. This is rarely a great disadvantage: although additional vertices might be needed to make a PLC ridge-protected, most of these vertices are probably needed to guarantee that the PLC has a CDT at all.

The running time of FLIPDELETEFACET is in $\mathcal{O}((m_{si} + m_{\text{left}} \cdot m_{\text{right}}\, m_v^{\lfloor d/2 \rfloor - 1}) \log m_v)$ in the worst case, and in $\Theta(m_v \log m_v)$ in the best case. The n_v term incurred by FLIPINSERTFACET is not incurred here because FLIPDELETEFACET presumably has access to pointers to the hyperfaces of T that represent f.

FLIPDELETEFACET is the first algorithm proposed for facet deletion in CDTs, short of retriangulating the entire PLC from scratch. One reason no previous algorithm exists is because it is difficult to quickly identify which d-simplices are no longer constrained regular when a facet is deleted (i.e. what is the shape of R?). A brute-force approach would compare every left vertex against every right d-simplex and vice versa, with visibility tests against every constraining hyperface of $R(\tau)$ (including all R's boundary simplices). This would take $\Omega(m_v^2 m_{si})$ time. FLIPDELETEFACET is usually substantially more efficient.

5. Symbolic Perturbations in the Flip Algorithms

Perturbations are an essential part of the flip algorithms because they ensure that no more than one minimal affinely dependent subset of the vertices lifts to a common non-vertical hyperplane at any one instant in time. Thus, there is a clear order in which bistellar flips can be performed without getting stuck. Note that any perturbation method is only effective if all geometric tests are robust against numerical error (e.g. done with exact arithmetic).

Let $X_0(\tau)$ be a linearly time-varying weighted PLC. $X_0(\tau)$ is geometrically identical to some base PLC X for every value of τ; only the vertex weights are different. Let W be a minimal affinely dependent subset of vertices of X. Because their lifted companions W^+ move along vertical linear trajectories, there are three possibilities for when W^+ can be affinely dependent (i.e. lie on a

common non-vertical hyperplane): never, at one time τ_W (at which time a flip might occur), or always (i.e. for all values of τ). The third possibility is eliminated by the perturbations. Call the second possibility a *flip opportunity*, denoted by γ_W.

The perturbation method, which adapts a suggestion of Edelsbrunner and Mücke [8, Section 5.4], visits the vertices one by one in an arbitrary sequence and perturbs the weight of each by a tiny quantity. The tiny quantities do not vary with τ. Intuitively, the first vertex weight is perturbed by an infinitesimal quantity, the second by the square of the infinitesimal quantity, the third by the cube, and so on. For the sake of mathematical rigor, the following method uses finite (but unspecified) perturbations. The goal is to separate the flip opportunities so no two occur at the same time, yet choose the perturbations small enough so the order in which two flip opportunities occur is not changed.

Let $X_i(\tau)$ be a time-varying PLC identical to $X_0(\tau)$, except that the first i vertex weights have fixed perturbations added. Each $X_{i+1}(\tau)$ is constructed from $X_i(\tau)$ according to the following rules. $X_{i+1}(\tau)$ is identical to $X_i(\tau)$ except that the weight of vertex $i+1$ is perturbed by a tiny (but nonzero) quantity chosen small enough that

- for any flip opportunities γ_V and γ_W with $\tau_V < \tau_W$ in X_i, in X_{i+1} the flip opportunity γ_V also precedes the flip opportunity γ_W; and
- for any flip opportunity γ_W in X_i, if $\tau_W < 0$, in X_{i+1} the flip opportunity γ_W also precedes time $\tau = 0$; whereas if $\tau_W > 0$, in X_{i+1} γ_W also follows time $\tau = 0$. This condition ensures that no $d+2$ vertices of $X_n(0)$ lift to a common non-vertical hyperplane, where $n = n_v$ is the number of vertices in X. Recall that FLIPINSERTFACET and FLIPDELETEFACET each stop or start at time $\tau = 0$.

The time at which a flip opportunity occurs varies linearly with the values of the weight perturbations (because the vertex weights vary linearly with time). Therefore, both conditions can always be satisfied by making the perturbation small enough. Observe that neither condition specifies what should happen if $\tau_V = \tau_W$ or $\tau_W = 0$ in X_i. In these cases, a perturbation might modify the flip opportunities so that $\tau_V \neq \tau_W$ and $\tau_W \neq 0$ in X_{i+1}—indeed, that is the ideal outcome. In the final PLC X_n, no two flip opportunities in X_n occur at the same time, nor does any occur at time $\tau = 0$.

In the flip algorithms in Sections 4, $X(\tau)$ should be everywhere replaced with $X_n(\tau)$.

The perturbations can be implemented symbolically without calculating the actual amount each vertex is perturbed. It is only necessary to know the order in which the vertex weights are perturbed, and the sign of each perturbation.

The geometric predicates that the flip algorithms use either compare the time when an event occurs with a fixed value, or compare the times when two events occur. The latter test is used heavily by the priority queue. There are several ways to incorporate perturbations into the events; the following is perhaps the easiest suggestion. Each event record on the queue stores the time at which the event occurs, but the time is not stored as a single number. Instead, each record stores the vertices of two d-simplices that share a hyperface that will lose its local regularity.

Comparing the times at which two events occur is equivalent to computing the sign of a polynomial over the vertex coordinates. The polynomial is linear in the vertex weights. If the polynomial is zero when the unperturbed vertex coordinates are substituted in, the two events occur simultaneously in the unperturbed model. Simulate the perturbations as follows to disambiguate the result of the test. Let V be the set of vertices whose weights appear in the polynomial. Choose whichever vertex of V comes first in the perturbation sequence, add an arbitrary constant with the right sign to its weight (for this test only), and recompute the polynomial. The magnitude of the constant is irrelevant (except for its effect on the running time of the algorithm that robustly computes the sign of the polynomial) because the polynomial is linear in each vertex weight. If the result is nonzero, return it, because the other perturbations in the sequence are much smaller and cannot reverse the sign.

If the polynomial still evaluates to zero, choose the vertex of V that comes second in the sequence, add a constant to its weight, and recompute the polynomial again. Repeat as necessary for each vertex in the sequence, but return the first nonzero result. If the final result is still zero after every vertex is perturbed, return zero.

The sequence of perturbations, and the sign of any particular vertex's perturbation, must remain consistent among all tests performed throughout the execution of an algorithm. Moreover, if a CDT provided as an input to a CDT update algorithm does not satisfy the nondegeneracy condition, it should have been computed using the same sequence and signs that the update algorithm uses.

How severe is this restriction? Every known CDT construction algorithm for $d > 2$, including the gift-wrapping and sweep algorithms, must use perturbations to guarantee its correctness when the nondegeneracy condition is not satisfied. Hence, it seems reasonable for any CDT update implementation to expect that the input CDT is consistent with some weight perturbation of the input PLC. Ideally, the implementation knows the identity of that perturbation.

What if the input to a CDT update implementation is a CDT that is not consistent with the perturbations the implementation uses? One can imagine "correcting" the input CDT so that it is consistent, but the input CDT and the corrected CDT could be almost entirely different, so correcting the input CDT might be as expensive as computing the corrected CDT from scratch. If the underlying PLC is not ridge-protected, it is possible that the PLC perturbed as the implementation prefers does not have a CDT at all.

An ambitious CDT update implementation can try to compute perturbations that are consistent with the input CDT. However, some valid CDTs are not consistent with *any* set of perturbations. (This is true even for unconstrained Delaunay triangulations.) But if the input CDT is consistent with a set of perturbations of the form described in this section, a simple greedy algorithm can find compatible perturbations. The greedy algorithm finds one vertex whose weight can be perturbed without causing any simplex of the CDT to stop being constrained semiregular, then finds another vertex that can be perturbed next with a smaller perturbation, and so on.

6. Simultaneous Vertex Insertion and Deletion

Facet insertion and deletion is not the only operation for which the techniques discussed here are useful. When dynamically updating CDTs, the ability to simultaneously insert or delete multiple vertices simultaneously may be needed. Consider the chicken-and-egg problem illustrated in Figure 11. An application wishes to insert a vertex v_1 into a segment s_1, but the presence of v_1 will cause a segment s_2 to no longer be regular, and the modified PLC will not have a CDT. This obstacle can be avoided by inserting a new vertex v_2 into s_2, thereby splitting s_2 into two subsegments that will both be regular. However, if v_2 is inserted first, s_1 will no longer be regular. The vertices v_1 and v_2 must be inserted simultaneously. This problem is motivated by a mesh generation algorithm for three-dimensional PLCs that have small angles, in which circumstances arise where arbitrarily many vertices must be inserted simultaneously to refine poor-quality elements [22].

Multiple vertices can be simultaneously inserted into (or deleted from) the segments of a ridge-protected three-dimensional CDT

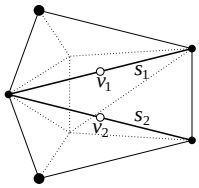

Figure 11: A constrained Delaunay tetrahedralization in which neither v_1 nor v_2 can be inserted without losing the constrained Delaunay property, unless they are inserted simultaneously. The constraining segments s_1 and s_2 are reflex edges of the polyhedron. Imagine the top and bottom vertices are slightly closer to you than the others.

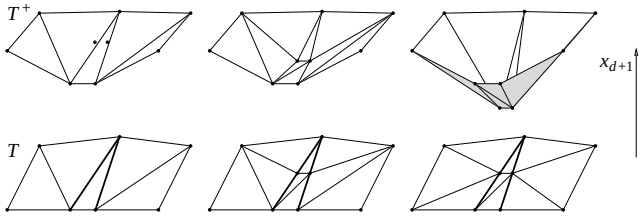

Figure 12: Simultaneous insertion of two vertices into two constraining segments. If one vertex were inserted first, the other segment would lose regularity. (In this two-dimensional example, the CDT can be saved by a refusal to flip the segments, but the algorithm is intended for three dimensions.)

by scheduling their weights so that each vertex bursts through the lifted segment in which it is inserted at approximately time $\tau = 0$ (subject to perturbations), and arrives at its final weight at exactly time $\tau = 1$, as Figure 12 illustrates. Details will appear in the full-length paper.

7. References

[1] Oswin Aichholzer, Franz Aurenhammer, and Hannes Krasser. *Pseudo-Triangulations from Surfaces and a Novel Type of Edge Flip*. Manuscript, 2003.

[2] Siu-Wing Cheng, Tamal K. Dey, Herbert Edelsbrunner, Michael A. Facello, and Shang-Hua Teng. *Sliver Exudation*. Journal of the ACM **47**(5):883–904, September 2000.

[3] L. Paul Chew. *Constrained Delaunay Triangulations*. Algorithmica **4**(1):97–108, 1989.

[4] ———. *Guaranteed-Quality Triangular Meshes*. Technical Report TR-89-983, Department of Computer Science, Cornell University, 1989.

[5] Kenneth L. Clarkson and Peter W. Shor. *Applications of Random Sampling in Computational Geometry, II*. Discrete & Computational Geometry **4**(1):387–421, 1989.

[6] Jesús A. de Loera, Francisco Santos, and Jorge Urrutia. *The Number of Geometric Bistellar Neighbors of a Triangulation*. Discrete & Computational Geometry **21**(1):131–142, 1999.

[7] Boris N. Delaunay. *Sur la Sphère Vide*. Izvestia Akademia Nauk SSSR, VII Seria, Otdelenie Matematicheskii i Estestvennyka Nauk **7**:793–800, 1934.

[8] Herbert Edelsbrunner and Ernst Peter Mücke. *Simulation of Simplicity: A Technique to Cope with Degenerate Cases in Geometric Algorithms*. ACM Transactions on Graphics **9**(1):66–104, 1990.

[9] Herbert Edelsbrunner and Raimund Seidel. *Voronoi Diagrams and Arrangements*. Discrete & Computational Geometry **1**:25–44, 1986.

[10] Herbert Edelsbrunner and N. R. Shah. *Incremental Topological Flipping Works for Regular Triangulations*. Algorithmica **15**(3):223–241, March 1996.

[11] Leonidas J. Guibas. *Kinetic Data Structures—A State of the Art Report*. Proceedings of the Third Workshop on Algorithmic Foundations of Robotics (Houston, Texas), pages 191–209, 1998.

[12] Barry Joe. *Three-Dimensional Triangulations from Local Transformations*. SIAM Journal on Scientific and Statistical Computing **10**:718–741, 1989.

[13] ———. *Construction of k-Dimensional Delaunay Triangulations using Local Transformations*. SIAM Journal on Scientific Computing **14**(6):1415–1436, November 1993.

[14] Charles L. Lawson. *Software for C^1 Surface Interpolation*. Mathematical Software III (John R. Rice, editor), pages 161–194. Academic Press, New York, 1977.

[15] ———. *Properties of n-Dimensional Triangulations*. Computer Aided Geometric Design **3**:231–246, 1986.

[16] Der-Tsai Lee and A. K. Lin. *Generalized Delaunay Triangulations for Planar Graphs*. Discrete & Computational Geometry **1**:201–217, 1986.

[17] Gary L. Miller, Dafna Talmor, Shang-Hua Teng, Noel Walkington, and Han Wang. *Control Volume Meshes using Sphere Packing: Generation, Refinement and Coarsening*. Fifth International Meshing Roundtable (Pittsburgh, Pennsylvania), pages 47–61, October 1996.

[18] Johann Radon. *Mengen Konvexer Körper, die Einen Gemeinschaftlichen Punkt Enthalten*. Mathematische Annalen **83**:113–115, 1921.

[19] Raimund Seidel. *The Upper Bound Theorem for Polytopes: An Easy Proof of Its Asymptotic Version*. Computational Geometry: Theory and Applications **5**:115–116, 1985.

[20] ———. *Constrained Delaunay Triangulations and Voronoi Diagrams with Obstacles*. 1978–1988 Ten Years IIG (H. S. Poingratz and W. Schinnerl, editors), pages 178–191. Institute for Information Processing, Graz University of Technology, 1988.

[21] Jonathan Richard Shewchuk. *A Condition Guaranteeing the Existence of Higher-Dimensional Constrained Delaunay Triangulations*. Proceedings of the Fourteenth Annual Symposium on Computational Geometry (Minneapolis, Minnesota), pages 76–85. Association for Computing Machinery, June 1998.

[22] ———. *Mesh Generation for Domains with Small Angles*. Proceedings of the Sixteenth Annual Symposium on Computational Geometry (Hong Kong), pages 1–10. Association for Computing Machinery, June 2000.

[23] ———. *Sweep Algorithms for Constructing Higher-Dimensional Constrained Delaunay Triangulations*. Proceedings of the Sixteenth Annual Symposium on Computational Geometry (Hong Kong), pages 350–359. Association for Computing Machinery, June 2000.

[24] ———. *Constrained Delaunay Tetrahedralizations and Provably Good Boundary Recovery*. Eleventh International Meshing Roundtable (Ithaca, New York), pages 193–204. Sandia National Laboratories, September 2002.

[25] ———. *What is a Good Linear Element? Interpolation, Conditioning, and Quality Measures*. Eleventh International Meshing Roundtable (Ithaca, New York), pages 115–126. Sandia National Laboratories, September 2002.

Appendix

A. Proof of Correctness of the Flip Algorithms

Note. *This section is not in the version of the paper published in the Nineteenth Annual Symposium on Computational Geometry.*

I begin with several basic combinatorial results about PLCs and CDTs that are needed for the algorithm proofs. They are presented without proof.

LEMMA 3. *Let p and q be two points in the triangulation domain of a PLC X, and suppose p and q can see each other. Suppose no constraining facet of X contains p. Say that p' is an ϵ-neighbor of p if $|pp'| \leq \epsilon$. Then there is a positive constant ϵ such that every ϵ-neighbor of p can see q.* ■

THEOREM 4. *Suppose that no $d+2$ vertices of a weighted PLC X lift to a common non-vertical hyperplane. If it exists, the weighted CDT of X is unique, and contains every constrained regular simplex of X.* ■

THEOREM 5 (DELAUNAY LEMMA). *Let X be a (weighted) PLC, and suppose no $d+2$ vertices of X lift to a common non-vertical hyperplane. A triangulation T whose vertices are the vertices of X is a (weighted) CDT of X if and only if T fills X, T respects X, and every hyperface of T is either locally regular or included in a constraining $(d-1)$-facet of X.* ■

The following theorem verifies a claim made in Section 3.

THEOREM 6. *Let X be a PLC, and let T be a triangulation that respects X. Let g be a non-constraining hyperface of T. Let W_R and W_C be the Radon subsets computed by the procedure* FLIP(T, g) *as discussed in Section 3, and let $W = W_R \cup W_C$. Note that $\mathrm{conv}(W_R)$ is a face of g (possibly g itself).*

If g is not locally regular, then $\mathrm{conv}(W_R)$ is neither regular nor constrained regular.

PROOF. Let s and t be the d-simplices that contain g. Let h_s be the unique hyperplane that contains s^+. As g is not locally regular, every vertex of s and t (including every vertex in W) lifts to a point in or below h_s.

The first claim is that $\mathrm{conv}(W_R)$ is not regular; suppose for the sake of contradiction that it is regular. Let h_R be a witness to its regularity. Every vertex of W_C lifts to a point above h_R.

W_R and W_C are a Radon partition of W. Let p be the Radon point $\mathrm{conv}(W_R) \cap \mathrm{conv}(W_C)$, which lies in the relative interior of $\mathrm{conv}(W)$. Treat the hyperplane h_s as a linear function that maps any point q in E^d to a height $h_s(q)$, and treat h_R similarly. Because h_s and h_R both include $\mathrm{conv}(W_R^+)$, $h_s(p) = h_R(p)$, as Figure 13 shows. However, $h_R(v) < h_s(v)$ for every vertex v of W_C. As p lies in the relative interior of $\mathrm{conv}(W_C)$, this is an impossibility. By contradiction, $\mathrm{conv}(W_R)$ is not regular. Nor is it constrained regular: every vertex in W is visible from p because s and t both contain p, and s and t respect X. ■

The following theorem ensures that the flip algorithms, applied to $X_n(\tau)$, never need to perform any retriangulation more complicated than a bistellar flip, and that no two flips happen simultaneously.

THEOREM 7. *Let V and W be two different minimal affinely dependent subsets of vertices in X. Let $V^+(\tau)$ be the set of lifted companions of the vertices in V, with heights assigned according to the perturbed PLC $X_n(\tau)$; and likewise let $W^+(\tau)$ be the set of lifted companions of the vertices in W. There is no time τ_* for which both of $V^+(\tau_*)$ and $W^+(\tau_*)$ are affinely dependent.*

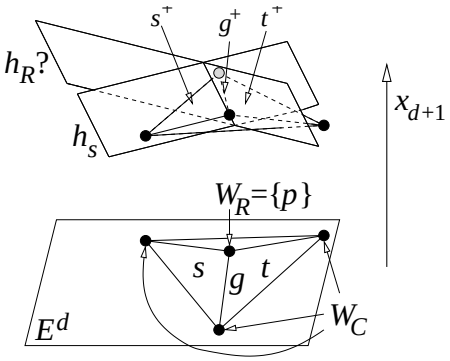

Figure 13: The hyperplanes h_s and h_R intersect over the Radon point p. If g is not locally regular, the simplex $\mathrm{conv}(W_R)$ cannot be regular because not every vertex of W_C can lift to a point above above h_R.

PROOF. Neither $V^+(0)$ nor $W^+(0)$ is affinely dependent by construction. As the weights vary linearly with time, each of $V^+(\tau)$ and $W^+(\tau)$ is affinely dependent for at most one value of τ.

Because each of V and W is minimal affinely dependent, each of them contains at least one vertex the other lacks. Let j be the index of the first vertex in the perturbation order that is in only one of the two sets. Suppose without loss of generality that vertex v_j is in V. The perturbation of the weight of v_j changes the time at which the flip opportunity γ_V occurs (because the time varies linearly with the weight), but not the time at which γ_W occurs. If the two times were equal prior to the perturbation, they are not equal for $X_j(\tau)$.

Next, suppose that for some i the flip opportunities γ_V and γ_W occur at different times in $X_{i-1}(\tau)$. Then, by the choice of perturbation quantities, the two flip opportunities occur at different times in $X_i(\tau)$. The result follows by induction on i. ■

The next lemma guarantees—under the right circumstances—that when a flip opportunity occurs, the flip can actually happen; the circumstances mentioned in Section 3 that might prevent a flip from being possible do not occur. In the following theorem and the rest of the appendix, $X(\tau)$ refers to the perturbed PLC $X_n(\tau)$.

LEMMA 8. *Suppose some triangulation T is the CDT of $X(\tau)$ for any $\tau_- < \tau < \tau_*$, where τ_- and τ_* are arbitrary real numbers. Suppose some hyperface g of T is no longer locally regular at time τ_*.*

Let s and t be the d-simplices that have g for a face. Let W be the minimal affinely dependent subset of the vertices of s and t (there is only one such subset). Let W_R and W_C be the Radon subsets of W computed by the procedure FLIP(T, g). *Suppose that no constraining facet of $X(\tau)$ includes $\mathrm{conv}(W_R)$.*

Then all of the d-simplices that FLIP *attempts to delete are present in T. Furthermore,* FLIP *transforms T into another triangulation T' such that T' is the CDT of $X(\tau)$ for any τ that satisfies $\tau_* < \tau < \tau_+$, where τ_+ is some number greater than τ_*.*

PROOF. Let W^+ be the set of lifted companions of the vertices in W, with heights assigned according to their weights in $X(\tau_*)$. Because τ_* is the first moment at which g is no longer locally regular, W^+ is minimal affinely dependent, just like W.

The vertices in W_C include the two vertices of s and t not shared with g. Let $T_R = \{\mathrm{conv}(W - \{v\}) : v \in W_C\}$. Two of the simplices in T_R—call them y_s and y_t—are faces of s and t. (If the flip is nondegenerate, $y_s = s$ and $y_t = t$.) One condition for the flip to be possible is that all the simplices in T_R must be present in T.

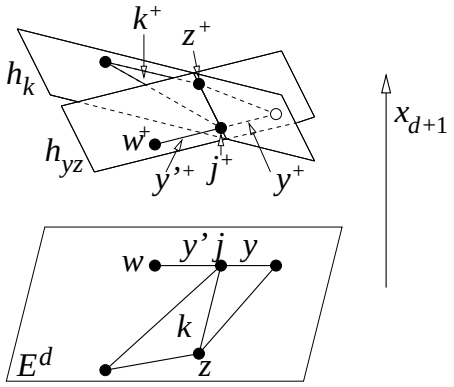

Figure 14: If k does not include w, then w^+ lies below k's witness hyperplane. Note that j and z may be simplices of any dimension. They are depicted here as points because of the limitations of paper.

Let j be the dimension of each simplex in T_R. Let y be any j-simplex in T_R that is also present in T. (There are at least two of these, namely y_s and y_t.) Let z be any $(d-j-1)$-simplex of T such that the d-simplex $\mathrm{conv}(y \cup z)$ appears in T. (Note that these steps of the proof echo Lines 19–20 of FLIP. If $j = d$, then z is the "empty simplex" and $\mathrm{conv}(y \cup z) = y$.) The goal of this proof is to show that for every such choice of z, and for every $y' \in T_R$, $\mathrm{conv}(y' \cup z)$ is a simplex of T, so the flip can be performed.

Because $\mathrm{conv}(y \cup z)$ is in the CDT T, it is constrained regular for any τ satisfying $\tau_- < \tau < \tau_*$; and as the motion of the lifted vertices is continuous, $\mathrm{conv}(y \cup z)$ is constrained semiregular at time τ_*. Let h_{yz} be the unique hyperplane in E^{d+1} that serves as a witness to the constrained semiregularity of $\mathrm{conv}(y \cup z)$. Because h_{yz} includes y^+, because y^+ has all but one of the vertices of W^+, and because W^+ is minimal affinely dependent at time τ_*, h_{yz} includes W^+. Other than the vertices of W and z, no vertex of $X(\tau_*)$ lifts to a point in h_{yz}, because if there were any other such vertex, it could be used to construct a subset of vertices that contradicts Theorem 7. Therefore, every vertex that is not in W nor z but is visible from $\mathrm{conv}(y \cup z)$ lifts to a point above h_{yz}.

Let y' be a simplex of T_R other than y. Let $j = y \cap y'$, and let w be the vertex of y' not shared by y; thus, $y' = \mathrm{conv}(j \cup w)$, as illustrated in Figure 14. Because $\mathrm{conv}(j \cup z)$ is a hyperface of the d-simplex $\mathrm{conv}(y \cup z)$, it is a face of T, and because T is a triangulation of X, T contains a second d-simplex k that includes $\mathrm{conv}(j \cup z)$. The present lemma claims that T contains $\mathrm{conv}(y' \cup z)$, or equivalently that $k = \mathrm{conv}(y' \cup z)$. Suppose for the sake of contradiction that k is some other simplex, as Figure 14 depicts. Then the apex of k is some vertex other than w, and therefore the apex lifts to a point above h_{yz}. It follows that w^+ (which lies in h_{yz}) lies below the witness hyperplane h_k of k. Because T is a CDT for $\tau_- < \tau < \tau_*$, k is constrained semiregular at time τ_*, so w is not visible from the relative interior of k. The next goal is to contradict this claim.

Let p be a point in the relative interior of $\mathrm{conv}(W_R)$. Because w and $\mathrm{conv}(W_R)$ are faces of y', and y' (being in T) respects X, w is visible from p; no facet of X can block the visibility. By assumption, no constraining facet includes $\mathrm{conv}(W_R)$, so no constraining facet contains p. By Lemma 3, there is a positive ϵ such that every ϵ-neighbor of p can see w. Recall that $\mathrm{conv}(W_R)$ is a face of every simplex in T_R, including y and y', so it is a face of k. Therefore, some point in the interior of k can see w, contradicting the conclusion of the previous paragraph.

Therefore, for every $y' \in T_R$, $\mathrm{conv}(y' \cup z)$ is a simplex of T. This argument holds for any valid choice of z, so every d-simplex that the FLIP procedure tries to remove is present in T, and FLIP will succeed.

Just prior to time τ_*, every hyperface of T is locally regular. At time τ_*, only hyperfaces of T that include $\mathrm{conv}(W_R)$ lose their local regularity. There are only two ways to triangulate the void left by the simplices removed by FLIP: with the original d-simplices $\{\mathrm{conv}(y' \cup z) : y' \in T_R, \mathrm{conv}(y \cup z) \in T\}$, or with the new d-simplices $\{\mathrm{conv}(y' \cup z) : y' \in T_C, \mathrm{conv}(y \cup z) \in T\}$. The hyperfaces between the new d-simplices are locally regular for all $\tau > \tau_*$. All the other non-constraining hyperfaces of the modified triangulation are still locally regular at time τ_* as a consequence of Theorem 7. They remain locally regular at least until the next flip opportunity occurs at some time τ_+. By the Delaunay Lemma (Theorem 5), the updated triangulation is the CDT of $X(\tau)$ for $\tau_* < \tau < \tau_+$. ■

The next several lemmata build a case for the correctness of FLIPINSERTFACET. A critical part is to show that FLIPINSERTFACET performs the flips necessary to maintain a CDT in the region R without getting stuck. (Recall that R is the union of the simplices of T that cross the inserted facet f.)

To aid the proof, let $R(\tau)$ be a time-varying PLC defined as follows. The vertices of $R(\tau)$ are the vertices of $X_n(\tau)$ that lie in R, and their time-varying weights are the same as in $X_n(\tau)$. $R(\tau)$ has one d-facet, namely R. The constraining facets of $R(\tau)$ are the simplices of T that either lie in the boundary of R, or lie in both R and a constraining facet of X. In summary, $R(\tau)$ is much like $X_n(\tau)$ except that everything outside the region R is omitted, and the boundary of R is represented as constraining facets.

The first lemma offers a constructive proof that $R(\tau)$ has a CDT for any $\tau \geq 0$—even if, for most values of $\tau > 0$, $X_n(\tau)$ does not have a CDT!

LEMMA 9. *Let X be a PLC that has a CDT T, and let f be a facet such that $X^f = X \cup \{f\}$ is a valid PLC. Let R be the union of all d-simplices of T that do not respect f. Let $X(\tau)$ be the time-varying PLC defined for use by* FLIPINSERTFACET *in Section 4, with weights perturbed as discussed in Section 5. Assume that T is consistent with the perturbations; that is, T is the CDT of $X_n(0)$. Let $R(\tau)$ be defined as above.*

Then $R(\tau)$ has a unique CDT for every $\tau \geq 0$, except for a finite number of values of τ during which a flip occurs. Furthermore, a call to FLIPINSERTFACET(X, T, f) *produces every such triangulation, in sequence as they are ordered by τ.*

PROOF. Consider the set of simplices in T that lie in the region R. If a simplex s (of any dimension) in the region R has at least one left vertex and one right vertex, then s is not a constraining simplex of X, because $X \cup \{f\}$ is a complex and the relative interior of s intersects f. If a simplex s does not have at least one left vertex and one right vertex, then s is constrained regular in $X_n(\tau)$ for all $\tau \geq 0$, as argued in Section 4. This includes every simplex that lies in the boundary of R. Likewise, every simplex that does not have both a left vertex and a right vertex is constrained regular in $R(\tau)$ for all $\tau \geq 0$.

The lemma is proved by induction on the sequence of flips performed by FLIPINSERTFACET. By assumption, R is a union of d-simplices that are constrained regular in X, so these d-simplices (and their faces) form a CDT of $R(0)$, and the lemma holds for any value of τ from zero until the first flip occurs.

Afterward, the algorithm maintains the following two invariants at any time τ between flips.

- The triangulation maintained by the algorithm is the CDT of $R(\tau)$.

- For every non-constraining hyperface f that will lose its local regularity at some time in the future, there is an event on the queue keyed by the time when the two d-simplices that currently include f will have the same witness hyperplane.

It is straightforward to verify that the second invariant is maintained by Lines 3–4 and 9–10 of FLIPINSERTFACET. The first invariant is proven as follows.

By Theorem 7, the CDT of $X_n(\tau)$ undergoes only one local retriangulation at any one time, and that retriangulation is always a bistellar flip (if a retriangulation is possible at all). Each call to FLIP computes a vertex set W_R, the vertices of the shared face that the flip tries to remove. By Theorem 6, $\mathrm{conv}(W_R)$ is no longer constrained regular at the time τ_* when the flip occurs. Therefore, $\mathrm{conv}(W_R)$ has at least one left vertex and one right vertex. The same is true of every simplex removed by the flip, because $\mathrm{conv}(W_R)$ is a face of every one of them. It follows that the algorithm never attempts to remove a boundary simplex of $R(\tau)$.

Because X^f is a complex, no constraining facet of X crosses the facet f, so no constraining facet of $R(\tau)$ crosses f. Because $\mathrm{conv}(W_R)$ crosses f and respects $R(\tau)$, $\mathrm{conv}(W_R)$ is not included in any constraining facet of $R(\tau)$. Therefore, Theorem 8 applies, and the new triangulation of R that the flip produces is the CDT of $R(\tau)$ for $\tau_* \leq \tau \leq \tau_+$, where τ_+ is the next time that a non-constraining hyperface loses its local regularity.

Because the invariants are maintained after each flip, the lemma follows by induction on the sequence of flips. ■

LEMMA 10. *Suppose that the PLCs X and $X^f = X \cup \{f\}$ (with their vertex weights perturbed as in $X_n(0)$) have CDTs. Let $R(\tau)$ be the time-varying PLC defined above. There is a constant τ_∞ such that the CDT of $R(\tau)$ respects f for any $\tau > \tau_\infty$.*

PROOF. By assumption, X^f has a CDT T^f which respects f. Let T_f be the subset of T^f containing the simplices included in f. By the definition of CDT, every simplex in T_f is constrained regular in the lowest-dimensional facet of X that contains it. That facet is either f or a face of f, so again by the definition of CDT, T_f is a CDT of f. The perturbation method described in Section 5 ensures that no $d+2$ vertices of X^f lift to a common non-vertical hyperplane, so T_f is the only CDT of f by Theorem 4.

The lemma follows because each simplex of T_f is constrained regular in $R(\tau)$ for some sufficiently large value of τ.

Let X_f be the $(d-1)$-dimensional PLC containing only f and its faces. Let s be any $(d-1)$-simplex in T_f. Because T_f is the CDT of f, there exists a witness h_s to the constrained regularity of s in X_f. Let τ_∞ be a time sufficiently large that no flip opportunities occur at or after time τ_∞. For $\tau \geq \tau_\infty$, s is constrained regular in $R(\tau)$ too, because one may construct a witness H_s to the constrained regularity of s in $R(\tau)$ as follows. Because $R(\tau)$ is d-dimensional, H_s is a d-dimensional hyperplane in E^{d+1}, whereas h_s is a $(d-1)$-dimensional hyperplane that lies in the affine subspace of E^{d+1} spanned by f and the x_{d+1}-axis. H_s is constructed by extending h_s in a direction affinely independent of this subspace, as illustrated in Figure 15. Choose this direction to be tilted so that the lifted companion of every left vertex lies above H_s. H_s might then be very steep, but it is not vertical, so if τ is sufficiently large, every right vertex lies above H_s too.

Thus for a large value of τ every left vertex and every right vertex lifts to a point above H_s, but what about a vertex v that is neither left nor right, but lies in f? Because s is constrained regular in X^f, either v is a vertex of s, v is not visible from the relative interior of s, or v's lifted companion v^+ lies above h_s.

Thus every vertex of $R(\tau_\infty)$ visible from the relative interior of s lifts to a point above H_s (except the vertices of s). Therefore, s is constrained regular in $R(\tau_\infty)$. ■

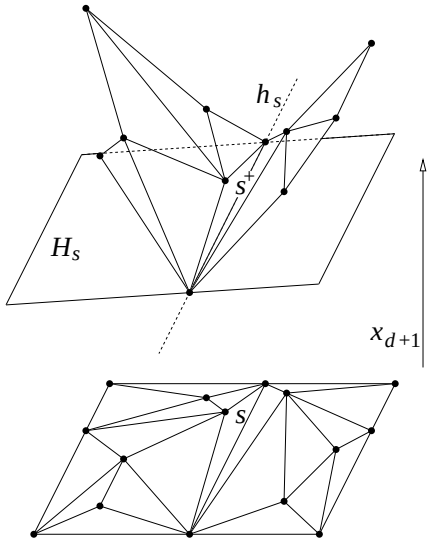

Figure 15: The $(d-1)$-dimensional hyperplane h_s contains the lifted constraining simplex s^+. (In a higher-dimensional picture, h_s would be a witness to the constrained regularity of s within some $(d-1)$-facet.) When τ is large enough, h_s can be extended by one dimension to become H_s, a witness to the constrained regularity of s in $R(\tau)$.

The main tool in proving the correctness of FLIPINSERTFACET is the Delaunay Lemma. Because all the non-constraining hyperfaces of the final triangulation are constrained regular, the final triangulation is a CDT of X^f.

THEOREM 11. *Suppose that the PLCs X and X^f have CDTs. Suppose T is a CDT of X, and is consistent with the perturbations; that is, T is the CDT of $X_n(0)$. Then the algorithm* FLIPINSERTFACET *returns a CDT of X^f. (This CDT is also consistent with the perturbations.)*

PROOF. Let T_∞ be the final triangulation produced by FLIPINSERTFACET. Every non-constraining hyperface of the initial triangulation T that lies outside the region R, or in the boundary of R, is still constrained regular after f is inserted. FLIPINSERTFACET does not modify any simplex outside of R or in the boundary of R, so these hyperfaces are all present in T_∞. Because they are constrained regular, they are locally regular in T_∞.

Let τ_∞ be a time chosen sufficiently large that no flip opportunities occur for any $\tau \geq \tau_\infty$. By Lemma 9, the simplices of T_∞ that lie in the region R form the CDT of $R(\tau_\infty)$. Therefore, every hyperface of T_∞ inside R is locally regular in $R(\tau_\infty)$. By Lemma 10, the CDT of $R(\tau)$ respects f, so these internal hyperfaces can be divided into two classes: those that lie in f, which are constraining simplices of X^f; and those that lie on one side of f.

Let g be any hyperface inside R but not included in f. Suppose without loss of generality g is to the left of f. Let s and t be the two d-simplices that share g. As s and t respect f, s and t have no right vertices. Because these vertices have the same weights in X^f as in $R(\tau_\infty)$, f is locally regular in X^f as well.

It follows that every non-constraining hyperface of T_∞ is locally regular. By the Delaunay Lemma (Theorem 5), T_∞ is the CDT of the perturbed X^f. Every simplex that is constrained regular in the perturbed X^f is constrained semiregular in the unperturbed X^f, so T_∞ is a CDT of the unperturbed X^f too. ■

THEOREM 12. *Suppose that the PLCs X and X^f have CDTs. Suppose T^f is a CDT of X^f, and is consistent with the perturbations. (In other words, if the vertex weights of X^f are perturbed to match the vertex weights of $X_n(0)$, T^f is still a CDT of X^f.) Then the algorithm* FLIPDELETEFACET *returns a CDT of X. (This CDT is also consistent with the perturbations.)*

PROOF. FLIPDELETEFACET maintains the following two invariants at any time τ between flips.

- The triangulation of R maintained by the algorithm is the CDT of $R(\tau)$, where the vertex weights of $R(\tau)$ follow the schedule for FLIPDELETEFACET discussed in Section 4.
- For every non-constraining hyperface that will lose its local regularity by time $\tau = 0$, there is an event on the queue keyed by the time when the two d-simplices that currently include the hyperface will have the same witness hyperplane.

It is straightforward to verify that the second invariant is maintained by Lines 1–2 and 7–9 of FLIPDELETEFACET. The first invariant is maintained because for any time τ at which FLIPINSERTFACET performs a flip, FLIPDELETEFACET performs the reverse flip at time $-\tau$. ■