International Journal of Computational Geometry & Applications

A POINT-PLACEMENT STRATEGY FOR CONFORMING DELAUNAY TETRAHEDRALIZATION[1]

MICHAEL MURPHY[2]
Department of Computer Science
University of Maryland—College Park,
College Park, MD 20742 [3]

and

DAVID M. MOUNT
Department of Computer Science and Institute for Advance Computer Studies
University of Maryland—College Park,
College Park, MD 20742 [4]

and

CARL W. GABLE
Geoanalysis Group, Earth and Environmental Science Division
Los Alamos National Laboratory
Los Alamos, NM 87544

Received (received date)
Revised (revised date)
Communicated by Editor's name

ABSTRACT

A strategy is presented to find a set of points that yields a Conforming Delaunay tetrahedralization of a three-dimensional Piecewise-Linear Complex (PLC). This algorithm is novel because it imposes no angle restrictions on the input PLC. In the process, an algorithm is described that computes a planar conforming Delaunay triangulation of a Planar Straight-Line Graph (PSLG) such that each triangle has a bounded circumradius, which may be of independent interest.

Keywords: Delaunay triangulations, Unstructured mesh generation

1. Introduction

In many two- and three-dimensional geometric modeling problems, notably the

[1] A preliminary version of this paper appeared in the Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms 2000, 67–74

[2] Author's Current Address: LizardTech, Inc. 821 Second Avenue, 18th Floor, Seattle, WA 98104

[3] Much of this work was done while the author was at Los Alamos National Laboratory.

[4] This author was supported by the National Science Foundation under grant CCR–9712379.

numerical approximation of the solution to a Partial Differential Equation with a Finite-Element type method,[21] it is very desirable to obtain a triangulation (tetrahedralization) that respects the domain of interest. The task of forming such decompositions, along with ensuring that the elements of the decompositions satisfy application-specific quality requirements, is sometimes referred to as unstructured mesh generation. See Ref. [2] for a survey. The Delaunay triangulation, a celebrated structure in Computational Geometry, can play a central role in this process[11,20] due to many important geometric properties and the existence of efficient algorithms to compute and maintain one with a dynamic set of points. (We shall assume the reader is familiar with the Delaunay triangulation and its basic properties, notably the "empty-circle" characterization. See Ref. [15] for a definition, a discussion of its properties, and algorithms for computing and maintaining the Delaunay triangulation.) Adapting the Delaunay triangulation, defined over point sets, to more complicated geometric domains such as arbitrary polygons or polyhedra, has proven to be a major challenge of unstructured mesh generation. To cope, researchers have developed the *Constrained Delaunay triangulation*, which changes the "empty-circle" criterion based on the domain. Another adaptation is the *Conforming Delaunay triangulation*, which is obtained when the domain is respected by the Delaunay triangulation of a set of representative points. Thus, to obtain a conforming Delaunay triangulation of a domain, the resolution typically must be increased through the addition of points, often called *Steiner points*. In the plane, these structures are well-understood and efficient algorithms to work with them are known. However, the analogs of these structures in three and higher dimensions pose many algorithmic challenges. In this paper, we address part of the challenges of three-dimensional unstructured mesh generation by giving a provably correct algorithm to construct a Conforming Delaunay tetrahedralization.

1.1. Piecewise-linear representations

We start by elaborating upon what we mean by a domain. We shall concentrate on piecewise-linear representations. Polygons and polyhedra fall into this class. However, for problems involving multiply connected boundaries, they are not expressive enough. For example, in a geological application, one may need to represent several layers of rocks, each having unique material properties that need to be distinguished in the simulation. The boundaries between rock layers, the *material interfaces*, may be very complicated, especially if there are cracks and faults present. For such demanding applications, the most general class of two-dimensional piecewise-linear representations, the *Planar Straight Line Graph* (PSLG), (see *e.g.* Ref. [15]) which encompasses polygons, polygons with holes, and all other planar, piecewise-linear, multi-material representations, is needed. PSLGs consist of vertices and line segments, also referred to as edges. Vertices are specified by providing the coordinates. Line segments are specified by giving the connections between vertices; the line segments must be non-overlapping, except when meeting at a common vertex. In three dimensions the *Piecewise-Linear Complex* (PLC) (using the notation in[14,20]) is the most general representation. In the PLC model,

the objects consist of vertices, line segments, and planar faces. Weiler[22] gives an equivalent formulation of a PLC as well as the "Radial-Edge" data structure to represent one.

1.2. Conforming Delaunay triangulations

We return to the problem of adapting Delaunay triangulations to piecewise-linear domains. One adaptation, the *Constrained Delaunay triangulation*, relaxes the "empty-circle" property. A formal definition and a $\Theta(n \log n)$ algorithm to compute one in the plane are given in Ref. [6]. Although in three dimensions, there is no immediate generalization of the Constrained Delaunay triangulation, a structure known as a *Conforming-Constrained Delaunay tetrahedralization* can be defined.[19] However, it should be noted that Constrained and Conforming-constrained Delaunay triangulations and tetrahedralizations are not as helpful with some numerical schemes because the quality requirements imposed on internal boundaries (material interfaces) imply that triangles incident upon these edges are "locally Delaunay." Specifically, for an internal edge in a PSLG, P, some schemes require that the two angles opposite that edge in the triangulation of P be nonobtuse. The generalization of these schemes to three dimensions often require that two tetrahedra sharing a face of a PLC contain their circumcenters.

Therefore, although no Steiner points are required to obtain a Constrained Delaunay triangulation and the Constrained-Conforming Delaunay tetrahedralization may require fewer Steiner points, algorithms to compute Conforming Delaunay triangulations and tetrahedralizations are of great interest in unstructured mesh generation. To obtain a Conforming Delaunay triangulation of a PSLG, P, one places Steiner points to ensure that all of the edges of P are represented in the Delaunay triangulation of the original point set together with the Steiner points. For this purpose, Steiner points never need to be placed anywhere but on the edges of the PSLG; this is often referred to as *edge refinement*. The sufficiency of edge refinement is a consequence of the following standard lemma (see *e.g.* Ref. [19]):

Lemma 1 *An edge e of a PSLG P with vertex set V is an edge of the Delaunay triangulation of V if and only if there exists a disk with the vertices of e on its boundary containing no points of V in its interior.*

Saalfeld's algorithm[17] for computing a Conforming Delaunay triangulation of a PSLG is based directly upon Lemma 1. To see the intuition, imagine for the moment that we have a PSLG, P, that consists of completely disjoint line segments. (That is, only one line segment is incident upon a vertex.) One strategy to satisfy Lemma 1 in this special case is to compute the closest distance between two segments $d_{min} > 0$. Next, pack a set of closed disks centered on every edge e with a radius strictly less than d_{min} so that two adjacent disks in the packing are tangent upon their common point of intersection with e. Steiner points can then be placed on e at these points of tangency, as shown in Figure 1(b). This strategy decomposes each edge of P into smaller edges, each of which satisfies Lemma 1 because the disks covering e cannot contain a point on another edge in their interior. Thus, we have a conforming Delaunay triangulation. Of course, many edges of a PSLG can

be incident on a common vertex. Hence, the first phase of Saalfeld's algorithm is to guard every vertex v by placing a sufficiently small disk centered around v so that the disk does not intersect any edge not incident upon v. The radius of such a disk could, for example, be computed by finding the minimum of the distance between the closest pair of vertices and the shortest distance between a vertex and a non-incident edge and dividing this quantity by 3. Steiner points are placed at the intersection of this disk with the edges of v. As shown in Figure 1(a), the minimum diameter circle passing through the portion of each edge from v to these Steiner points will be empty, satisfying Lemma 1. After this step, what remains is in essence a set of disjoint edges that can be processed in the manner just described.

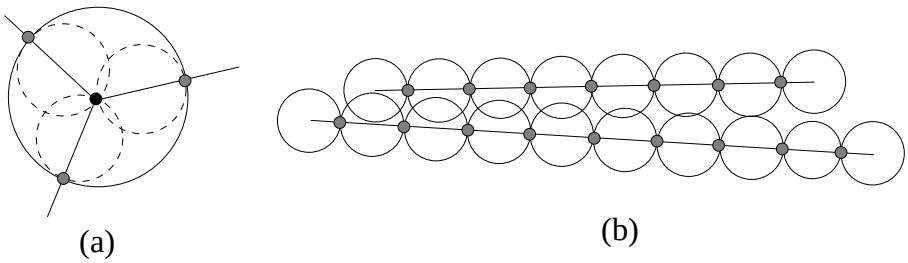

Figure 1: Key steps in Saalfeld's Conforming Delaunay triangulation algorithm. **(a):** Protecting the vertices. **(b):** Covering the edges with empty tangent circles.

Because it can place an excessive number of Steiner points, Saalfeld's algorithm should be viewed more as a simple existence proof of a Conforming Delaunay triangulation than as a practical algorithm. There are at least two noteworthy provably correct algorithms to find a Conforming Delaunay triangulation of a PSLG that are sensitive to the number of Steiner points. The first is Edelsbrunner and Tan's algorithm,[9] which gives a striking $O(n^3)$ combinatorial upper bound on the number of Steiner points placed, where n is the input size. The other is Ruppert's Delaunay-Refinement algorithm[16] using "the Quitter"[18] to resolve small input angles. Although the latter algorithm does not admit combinatorial bounds on the number of Steiner points placed – the bounds come from the local-feature-size, an intrinsic geometric property of the domain – Ruppert's algorithm is quite practical and can be used to construct a no-small-angle triangulation, useful in bounding discretization error in the Finite-Element method.

1.3. A strategy for conforming Delaunay tetrahedralization

Many heuristic, or at least unproven, algorithms can be found in the literature to successfully compute a Conforming Delaunay tetrahedralization. See *e.g.* Ref. [12]. However, a provably correct algorithm for computing a Conforming Delaunay tetrahedralization of a general PLC is an open problem.[3,5,4,10,1] Some provably correct algorithms to find a conforming Delaunay tetrahedralization make stringent angle restrictions on the input PLC.[20,14] To be sure, these algorithms

were designed to provide "quality" Delaunay tetrahedralizations, where a bounded ratio of circumradius to shortest edge of each tetrahedron is the measure of interest, rather than any Delaunay tetrahedralization. However, unlike Ruppert's planar algorithm, which has a similar motivation and makes similar angle restrictions that can be side-stepped to obtain a conforming Delaunay triangulation (albeit with no no-small-angle quality guarantees in the vicinity of the small angles), these restrictions are not as readily resolved. Our purpose is to give an algorithm to find a Conforming Delaunay triangulation of a PLC P with no restrictions on the angles of P, a step towards both practical and provably correct Delaunay-based mesh generation in three dimensions.

The problem in three dimensions is more involved because both edges *and* faces of a PLC must be refined until they are part of the Delaunay tetrahedralization of the augmented point set. The three-dimensional analog of Lemma 1 for edges in a PLC not part of any face remains the same except that we require empty balls rather than disks. A straightforward generalization of Saalfeld's algorithm can be used to process these edges. Therefore, such hanging edges are considered no further. Rather, we are concerned with refining the faces. The analog of Lemma 1 becomes:

Lemma 2 *A triangular face f (or a face with four or more cocircular vertices) of a PLC P with vertex set V is a face in the Delaunay tetrahedralization of V if and only if there exists a ball with the vertices of f on its boundary containing no points of V in its interior.*

The algorithm we describe is motivated by the following observation: Suppose we are given a set of disjoint faces in $\mathbb{R}^3$ which we wish to refine so that the Delaunay tetrahedralization of the augmented point set conforms to these faces. A sufficient but not necessary condition is to find a *planar* Delaunay triangulation of each face with the property that for each triangle, t, the circumscribing sphere of radius equal to the radius of the (planar) circumcircle of t does not intersect any other face. Note that this sphere is the one of minimum radius circumscribing t. A planar conforming Delaunay triangulation of each face where each triangle has a radius bounded by the distance to the nearest face in the PLC will satisfy this condition. To obtain this triangulation, Steiner points may have to be placed along the edges of a face as well as in its interior. We give an adaptation of Chew's guaranteed-quality Delaunay triangulation algorithm[7] for this purpose in Section 2.

Of course, the faces need not be disjoint. As a consequence, the above strategy fails because the distance between two incident faces is zero. However, methods used in our adaptation of Chew's algorithm to protect vertices and edges extend to three dimensions. What remains after these protection phases is a set of disjoint subfaces which we can refine with our bounded circumradius conforming Delaunay triangulation algorithm. We show that the results of the protection phases and the planar triangulation phases do not interfere. Thus, we have a refinement of the PLC such that a Conforming Delaunay tetrahedralization can be obtained from its vertices.

2. Delaunay triangulations with bounded circumradii

We proceed by finding a conforming Delaunay triangulation of a PSLG such that the circumradius of each triangle is bounded from above by a pre-specified constant. One early guaranteed-quality Delaunay triangulation algorithm due to Chew[7] shows promise. Although the intention of his algorithm is to produce a Constrained Delaunay triangulation such that the angles in the triangulation are between 30 and 120 degrees, it also generates triangles with bounded circumradii. However, before applying his algorithm to our task, two problems need to be addressed. First, the precondition of his algorithm requires that no input angle be less than 30 degrees. Second, his algorithm does not guarantee a Conforming Delaunay triangulation, only a Constrained Delaunay triangulation. We present modifications that address both issues while maintaining the user-specified upper bound on the largest circumradius.

2.1. Review of Chew's algorithm

Chew's algorithm takes as input a PSLG P such that no angle incident upon a vertex is less than 30 degrees and a parameter r_{max} from the user. The output is a Constrained Delaunay triangulation such that the circumradius of each triangle does not exceed r_{max}. The first step of his algorithm refines the edges of P into subsegments whose lengths are in the range $[h, \sqrt{3}h]$ for some $h \leq r_{max}$. The parameter h must be chosen small enough so that such a refinement is possible and so that h is no larger than the closest distance between any two (Steiner or input) vertices. Because of the precondition on the smallest angle, such a value always exists. (A general strategy for finding such an edge partition is given below.) After computing the Constrained Delaunay triangulation of the modified PSLG, circumcenters of triangles whose radii are larger than h are inserted, one at a time. The Constrained Delaunay triangulation can be restored after each such Steiner point insertion using Lawson's algorithm.[13] The process continues until no triangles with circumradii exceeding h exist, which Chew demonstrates always occurs eventually.

2.2. Treating the small angles

If the input PSLG contains angles less than 30 degrees, finding a value for h so that the PSLG is decomposed into edges of length in the range $[h, \sqrt{3}h]$ and so that no two vertices are of distance less than h is impossible. However, recall that Chew's primary goal was not to assert that his algorithm creates triangles with small circumradii but rather to bound the angles of each triangle. Although we make use of the bounds on the edge lengths in our proof, we can tolerate arbitrarily small angles by transforming PSLGs containing them to ones Chew's algorithm can process by adding a vertex-protection phase, resembling Saalfeld's, prior to invoking Chew's algorithm. We describe this phase in conjunction with the initial edge-refinement process.

Consider a PSLG P. Let δ be the minimum distance either between any two vertices or between a vertex and any non-incident edge. Let ϕ be the minimum of

$\pi/4$ and the angle between two edges that share a vertex. Let $r = \delta/3$, and let h be any quantity that is no greater than the length of a chord of a circle of radius r subtending an angle of $\phi/2$. (By the law of cosines, $h \leq r\sqrt{2(1-\cos(\phi/2))}$.) Because $\phi \leq \pi/4$, it is easy to verify that $h \leq r/2$; see Figure 2.

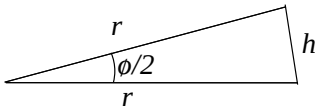

Figure 2: If $\phi \leq \pi/4$ then $h \leq r/2$.

As in Saalfeld's vertex-protection phase, each vertex is surrounded by a protecting circle of radius r. Steiner points are placed at the intersection of each circle with the edges of P. This subdivides each circle into a collection of arcs of angular sizes at least ϕ. Each arc of angle θ is further subdivided into $k = \lfloor 2\theta/\phi \rfloor$ subarcs of equal sizes. Of course, if the subarc is outside the external boundary, we do no need to refine it. Steiner points are placed at the endpoints of these intervals. By connecting consecutive Steiner points, we form a convex polygon surrounding each input vertex. We shall refer to the edges of this polygon as its *rim edges*. We connect the central vertex to these points by a set of *spokes*, forming a set of isosceles triangles. This is illustrated in Figure 3 for an internal vertex in the domain.

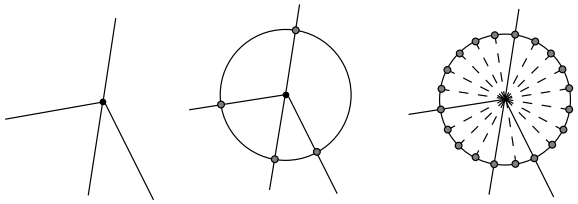

Figure 3: A vertex of a PSLG is protected by a circle of appropriate radius. Intersection points are added, and Steiner points are placed on the circle to satisfy Chew's point-spacing criteria.

We assert that the lengths of rim edges are in the interval $[h, \sqrt{3}h]$. This is true because $\theta \geq \phi$ implies that $k \geq 2$. Combining this and the definition of k we have

$$\begin{aligned}
\frac{k\phi}{2} &\leq \theta < \frac{(k+1)\phi}{2} \\
\frac{\phi}{2} &\leq \frac{\theta}{k} < \frac{(k+1)\phi}{2k} \\
\frac{\phi}{2} &\leq \frac{\theta}{k} < \frac{3\phi}{4} < \sqrt{3}\frac{\phi}{2}.
\end{aligned}$$

Each subarc is of angular size θ/k. Two points subtending a subarc of angle $\phi/2$ are at distance h apart, and two points subtending a subarc of $\sqrt{3}\phi/2$ are at most distance $h\sqrt{3}$ apart, as desired.

As we have just seen, the angles between consecutive spokes is less than $3\phi/4 < \pi/3$. Consider any isosceles triangle whose base is of length b and whose opposite angle at most $\pi/3$. It is straightforward to show that its circumcircle extends a distance at most $b/(2\sqrt{3})$ beyond the base. Since $b \leq \sqrt{3}h$, the circumcircles from this part of the construction do not extend a distance more than $h/2$ outside of the surrounding convex polygon. We use this fact later.

Observe that no two non-incident pairs (vertex-vertex or vertex-edge) can be closer than distance $3r$, implying that even after adding the surrounding circles of radius r, all non-incident entities are separated by a distance of at least $r > 2h$.

We create a new PSLG P' to be supplied to Chew's algorithm by starting with P, adding the protecting polygons, discarding the original vertices of P and the portion of its edges that fall inside the protecting polygons. The edges of P' which are not rim edges are of length at least r (the minimum distance between circles). Each such edge of length l is subdivided into $j = \lfloor l/h \rfloor$ subsegments of equal lengths and Steiner points are placed at the endpoints of each subsegment. As noted earlier, $h/2 \leq r \leq l$, implying that $j \geq 2$. By the same argument above, it follows that the length of each subsegment is in the interval $[h, h\sqrt{3}]$. Because no two consecutive vertices are closer than h, and no two non-incident entities are closer than $2h$, it follows that no two vertices of this construction are closer than h. Thus, we can apply Chew's algorithm to the result.

We claim that after applying Chew's algorithm to P' and restoring the vertices of P and the spokes of the protecting polygons, any pair of triangles sharing a rim edge (*i.e.*, one spoke triangle and one generated by Chew's algorithm opposite to it) are "locally Delaunay." The reason is that Chew's algorithm does not place any vertices within distance $h/2$ of any boundary edge. By the observation made earlier, the circumcircles defined by these isosceles triangles do not extend outside of the convex polygon by a distance more than $h/2$. Thus Chew's triangles are protected from the circumcircles of the "spoke triangles." As a check, the circumcircles generated by Chew's algorithm do not penetrate the protecting polygon by a distance of more than $3h/2$ and therefore cannot contain the vertex of P protected by the polygon. Thus, the spoke triangles and Chew's triangles do not interfere with each other.

2.3. Protecting the input edges

However, running the above vertex-protection scheme followed by Chew's algorithm on the modified PSLG P' will not necessarily give a Conforming Delaunay triangulation of P'. This is because a pair of Chew's triangles sharing a boundary edge can be obtuse, violating the "empty-circle" condition (although they are "Constrained Delaunay.") See Figure 4.

To remedy this, we would like to buffer each edge e so that a circumcircle of a triangle generated by Chew's algorithm on one side of e cannot contain a vertex on the opposite side of e. The strategy is to extrude parallel edges from e into the domain. Extrusion can begin after the protecting circles of radius r for each vertex have been computed as above. Again, let ϕ be the minimum of $\pi/4$ and the angle between two edges that share a vertex in P. The edge extrusion distance, d_e, can

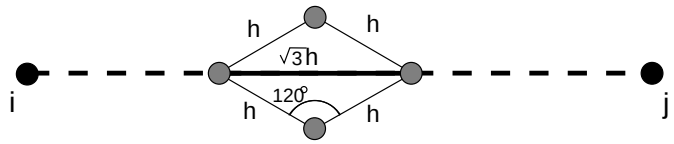

Figure 4: Why Chew's algorithm only guarantees a Constrained Delaunay triangulation. Assume $e = (i, j)$ is an internal boundary edge and the points on opposite sides were placed by Chew's algorithm.

be any quantity that is no greater than the length of a chord of a circle of radius r subtending an angle of $\phi/3$. For each edge $e = (i, j)$ of P, we place Steiner points on the protecting circles of i and j at a distance d_e from e inside the domain. (If e is an internal edge, this will result in four Steiner points; if e is an external boundary edge, this will give only two points.) We form the protecting edges parallel to e by connecting these Steiner points to their counterparts on the circle protecting the adjacent vertex, as illustrated in Figure 6(b). Note that using $\phi/3$ to compute d_e means that two extruded edges are never closer than a distance of d_e.

We now consider the initial *buffer zone* of a face to be anything inside the protecting circles or within a distance d_e of an edge of the face. For the purposes of invoking Chew's algorithm, we need to refine the boundary of all initial buffer zones into segments of length $[h, h\sqrt{3}]$ such that no two points are closer than h for some suitable value of h. Specifically, it is necessary to ensure that $h \leq 2d_e$ or the buffer zones will not be large enough to prevent interference between triangles generated by Chew's algorithm on opposite faces of P. However, this constraint is trumped by the constraint that h must be chosen such that no two points are closer than h after refinement; since the closest pair of points can be of distance d_e (*i.e.,* the endpoints of two extruded edges on the same protecting circle), it is necessary that $h \leq d_e$. We claim it is possible to refine the boundary of the buffer zone using any $h \leq d_e/2$ to satisfy Chew's length requirements above because it is impossible to refine the boundary to create angles less than 30 degrees when viewed from outside the buffer zone. For reasons that will become apparent below, it is desirable to refine the edge protectors and e identically. This alignment of Steiner points is illustrated in Figure 5; the refinement of the initial buffer zone is shown in Figure 6(c).

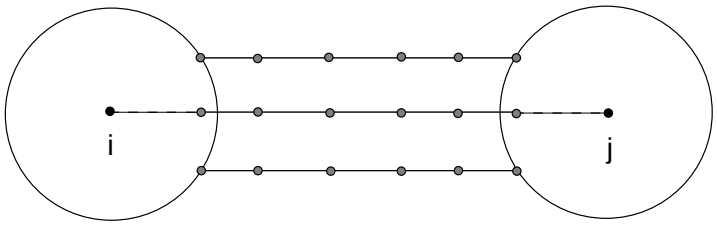

Figure 5: Two parallel edges are extruded from $e = (i, j)$ to connect i and j's protecting circles. The three edges are refined with the same point distribution.

After invoking Chew's algorithm on this modified PSLG, we triangulate the buffer zone, as illustrated in Figure 6(d). We now argue that these triangles do not interfere with the triangles generated by Chew's algorithm. For the same reasons as before, the spoke triangles are locally Delaunay. The other type of triangles, those incident upon Steiner points placed on e, cannot extend a distance more than $h/2$ outside of the buffer zone. This is because of the relationship between h and d_e. Specifically these triangles are the result of (arbitrary) triangulations of rectangles of dimensions at most $\sqrt{3}h$ by d_e and $d_e \geq h$ by definition.

We now have the following theorem:

Theorem 1 *A PSLG can be refined so that a Conforming Delaunay triangulation is obtained with the property that the circumradii of each Delaunay triangle is bounded from above by a pre-specified constant using a finite number of Steiner points.*

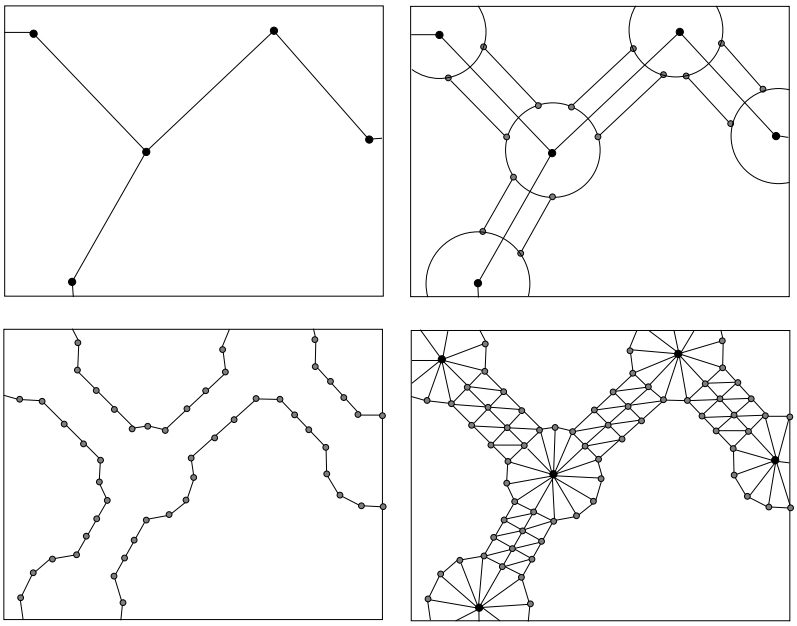

Figure 6: **(a)** A portion of a PSLG **(b)** Finding protecting circles and edges. **(c)** The PSLG passed to Chew's algorithm to triangulate everything outside the buffer zones. **(d)** The triangulation of the interior of the buffer zone.

3. Extending to three dimensions

3.1. Overview

We extend the above algorithm to find a Conforming Delaunay tetrahedralization of a PLC, P. Like the two-dimensional algorithm, we create buffer zones around the shared vertices and edges of faces. We then use Chew's algorithm to create a (planar) conforming Delaunay triangulation of the portions of the faces

outside the buffer zones such that the minimum diameter sphere of every triangle in these triangulations cannot intersect another planar face outside a buffer zone. This is illustrated in Figure 7. This satisfies Lemma 2 for each triangle if the buffer zones do not contain points where these spheres intersect. Again, the crux is to define the buffer zones and refine them into triangles so that they do not interfere with the triangles produced by Chew's algorithm. Fortunately, the vertex and edge protection methods of our planar bounded-radius conforming Delaunay triangulation algorithm given above extend easily into three dimensions. The intuition behind the extensions is that when we protect shared vertices of P, we use protecting spheres centered at these vertices rather than protecting circles. To protect edges, we place protecting cylinders around them. However, the phrases "protecting spheres"and "protecting cylinders" are not entirely accurate because it is acceptable for a Delaunay edge to pierce these spheres and cylinders in the final tetrahedralization of the point set, if the piercing edge does not cross a face of P.

3.2. Creating the buffer zones

The first step is to compute the size of the buffer zones. The values of d_e and r defined above can be used for this purpose. The only difference in their computation is that we use as our δ the minimum distance between any two adjacent vertices or any two non-incident entities (vertex-vertex vertex-edge, vertex-face, edge-edge, edge-face, face-face). To avoid problems with ambiguities caused by two incident faces that pass close to each other far from the points of incidence (as can occur with non-convex faces), assume all faces are triangulated arbitrarily. We do not take into account dihedral angles at this point.

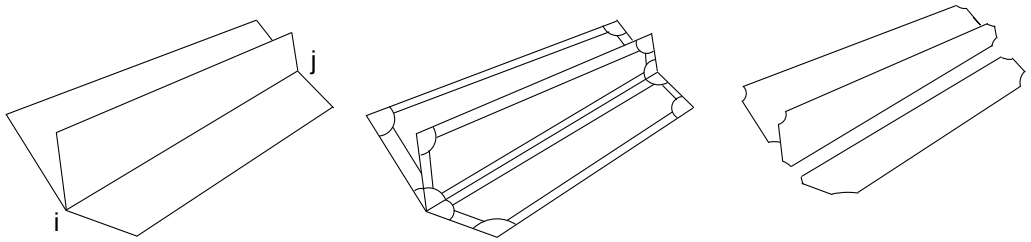

Figure 7: **(a)** Edge (i, j) has three faces incident upon it. **(b)** The buffer zones on each face formed by vertex-protection and edge-extrusion. **(c)** The disjoint faces passed to Chew's algorithm to triangulate.

For each vertex v, intersect each face incident upon v with a protecting circle of radius r centered at v. This could be viewed as forming a protecting sphere of radius r around v. We also install the edge-protectors. Consider an edge e. For every face incident upon e, we place a Steiner point on the protecting arcs of that face at both of the endpoints of e at a distance d_e from e and form the parallel protecting edges as before by connecting these Steiner points. This can be imagined by intersecting a cylinder with radius d_e and axis e with the incident faces of e. We also place Steiner points at the orthogonal projection of these Steiner points onto e. As before, we do

not consider the arcs on each face inside the edge protector; we consider the buffer zone of a face to be anything inside the protecting circle or within a distance d_e of e on the face.

We now refine the boundaries of the buffer zones. To do so, we compute a spacing value h for Chew's algorithm based upon the closest pair of features. This is a function of the dihedral angles and the closest pair of Steiner points placed in the vertex and edge protection steps above. Specifically, let α be the smallest dihedral angle of P. Let d_f be the length of the base of an isosceles triangle with angle α and side lengths d_e. We claim any $h \leq \min(d_e, d_f)/2$ can be used to refine the buffer zone of each face into segments of length between $[h, h\sqrt{3}]$ such that no two Steiner points after the refinement of each face will be closer than h. As before, we must ensure that the protecting edges are refined identically, as well as the edges they are protecting. We can now run Chew's algorithm on the portion of each face of P outside of the buffer zones to create a conforming Delaunay triangulation with h as the maximum radius of any circumcircle.

We claim that every one of Chew's triangles satisfies Lemma 2. First, spheres of radii equal to the circumradii of Chew's triangles not incident upon an edge of the buffer zone cannot intersect any other entity in the PLC, due to the bounds on the circumradii given by h. Second, if a circumradius of one of Chew's triangles intersects another face, it is because it is incident upon the buffer zone. We claim it only intersects a portion of a buffer zone where no points are placed. This follows from the alignment of Steiner points of the protecting edges with the input edges and the bounds on the circumradii. It is straightforward to show the triangles inside the buffer zones do not interfere with each other because of the spherical and cylindrical distributions of the points, point alignment on the edges and edge-protectors, and edge spacing. This gives our main result:

Theorem 2 *A PLC P can be refined so that the Delaunay tetrahedralization of its augmented vertex set conforms to P with a finite number of Steiner points.*

4. Conclusion

In essence, we have given a generalization of Saalfeld's Conforming Delaunay triangulation algorithm to three dimensions. Like its planar counterpart, our algorithm places far too many points to be practical and should be viewed more as an existence proof. Indeed, we chose to mimic Saalfeld's algorithm because it was the most straightforward one we knew. Nonetheless, we conjecture that the techniques we present for protecting vertices and edges in the presence of small input angles can be adapted to work in practice (*e.g*, by allowing r and d_e to vary according to some notion of "local feature size" and application-driven edge length requirements) with a Delaunay refinement algorithm which is stingier with Steiner points.

Finally, we would like to add some remarks concerning mesh quality. Since we have chosen to use Chew's algorithm, the quality of the boundary triangles generated by his algorithm is guaranteed in both a no-small-angle as well as a no-large-angle sense. This is a good starting point for generating tetrahedra with bounded radius-to-edge ratios. Moreover, the triangles inside the buffer zones are

all nonobtuse. Finally, because we have ensured that the minimum diameter circumsphere of each boundary triangle is empty, meshes generated with our strategy would guarantee an M-matrix when Poisson's equation is discretized using the popular Control-Volume method.[14,8] Satisfying this latter criterion is very desirable, especially in time-dependent simulations.

References

1. T. Baker. Delaunay-Voronoï methods. In J. F. Thompson, B.K Soni, and N.P. Weatherill, editors, *Handbook of Grid Generation*, pages 16.1–16.11. CRC Press, Boca Raton, 1999.
2. M. Bern and D. Eppstein. Mesh generation and optimal triangulation. In D.-Z. Du and F. K. Hwang, editors, *Computing in Euclidean Geometry*, volume 1 of *Lecture Notes Series on Computing*, pages 23–90. World Scientific, Singapore, 1992.
3. M. Bern and P. Plassmann. Mesh generation. *Unpublished Manuscript*, 1997. Available at http://www.ics.uci.edu/~eppstein/280g.
4. P. R. Cavalcanti and U. Mello. Three-dimensional constrained Delaunay triangulation: A minimalist approach. In *Proc. 8th International Meshing Roundtable*, Lake Tahoe, CA, 1999. Sandia National Laboratories.
5. S.W. Cheng, T.K. Dey, H. Edelsbrunner, M.A. Facello, and S.H. Teng. Sliver exudation. In *Proc. 15th Annu. ACM Sympos. Comput. Geom.*, 1999.
6. L. P. Chew. Constrained Delaunay triangulations. *Algorithmica*, 4:97–108, 1989.
7. L. P. Chew. Guaranteed-quality triangular meshes. Technical Report TR 89-983, Dept. Comput. Sci., Cornell Univ., Ithaca, NY, 1989.
8. Q. Du, V. Faber, and M. Gunzburger. Centroidal Voronoi tessellations: applications and algorithms. *SIAM Review*, 41:637–676, 1999.
9. H. Edelsbrunner and T.-S. Tan. An upper bound for conforming Delaunay triangulations. *Discrete Comput. Geom.*, 10(2):197–213, 1993.
10. P. L. George. Tet meshing: Construction, optimization, and adaptation. In *Proc. 8th International Meshing Roundtable*, Lake Tahoe, CA, 1999. Sandia National Laboratories.
11. P. L. George and H. Borouchaki. *Delaunay Triangulation and Meshing: Application to Finite-Elements.* Hermes, New York, NY, 1998.
12. M. Held, J. T. Klowsowki, and J. S. B. Mitchell. Evaluation of collision detection methods for virtual reality fly-throughs. In C. Gold and J.-M. Robert, editors, *Proc. 7th Canad. Conf. Computat. Geometry*, pages 205–210, 1995.
13. C. L. Lawson. Software for C^1 surface interpolation. In J. R. Rice, editor, *Math. Software III*, pages 161–194, New York, NY, 1977. Academic Press.
14. G. L. Miller, D. Talmor, S.-H. Teng, N. Walkington, and H. Wang. Control volume meshes using sphere packing: Generation, refinement, and coarsening. In *Proc. 5th International Meshing Roundtable*, Albuquerque, NM, 1996. Sandia National Laboratories.
15. F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, New York, NY, 1985.
16. J. Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *Journal of Algorithms*, 18(3):548–585, 1995.
17. A. Saalfeld. Delaunay edge refinements. In *Proc. 3rd Canadian Conf. Comp.*

Geometry, pages 33–36, 1991.

18. J. R. Shewchuk. *Delaunay refinement mesh generation.* PhD thesis, School of Computer Science, Carnegie Mellon University, 1997. Available at http://www.cs.berkeley.edu/~jrs.
19. J. R. Shewchuk. A condition guaranteeing the existence of higher-dimensional constrained Delaunay triangulations. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, 1998.
20. J. R. Shewchuk. Tetrahedral mesh generation by Delaunay refinement. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, 1998.
21. G. J. Strang and G. Fix. *An Analysis of the Finite-Element Method.* Prentice–Hall, 1973.
22. K. J. Weiler. The radial edge structure: A topological representation for non-manifold geometric modeling. In J. Encarnacao M. Wozny, H. McLaughlin, editor, *Geometric Modeling for CAD Applications.* Springer Verlag, 1987.