Chapter 14

Meshing smooth surfaces and volumes

The theory of surface sampling and restricted Delaunay triangulations developed in the last two chapters seems to mark a clear path to designing a Delaunay refinement algorithm for triangular mesh generation on a smooth surface: maintain a restricted Delaunay triangulation by maintaining a Delaunay tetrahedralization, refine it by inserting new vertices at the centers of circumballs of restricted Delaunay triangles, and continue refining until the sample is dense enough to guarantee topological correctness, geometric accuracy, and high triangle quality. Upon termination, the algorithm returns a mesh that is related to the input surface by an isotopy and enjoys all the geometric guarantees offered by the Surface Discretization Theorem (Theorem 13.22).

The fly in the ointment is that it is very difficult to know when the algorithm has succeeded and can stop refining. In theory, we can achieve these guarantees by generating a 0.08-sample or making sure that every restricted Voronoi cell is 0.09-small. In practice, there are two problems. First, it is both difficult and expensive to compute the local feature size function. The Delaunay refinement algorithms we have studied for polygonal and polyhedral domains use the local feature size in their analysis, but do not need to compute it. Curved domains are different: without computing the local feature size, it is difficult to be certain that a mesh generator has not overlooked some high-curvature feature of the domain. Second, in practice it is usually possible to recover a surface with a vertex set much less dense than a 0.08-sample, and we would prefer to have a mechanism for knowing when we can stop early.

The first surface meshing algorithm we present assumes that the user can somehow compute the LFS function, has domain knowledge that makes it possible to specify a lower bound on the LFS function, or simply wants triangles small enough that resolution is not an issue. If a mesh of uniformly-sized elements is acceptable, a constant lower bound on the feature size will do.

For many applications, however, no such approximation is known, or the concomitant over-refinement is unacceptable. Fortunately, we can appeal directly to the Topological Ball Theorem (Theorem 13.1), which says that if the faces in $\mathrm{Vor}\, S$ intersect Σ in topological closed balls of appropriate dimensions, then the underlying space of $\mathrm{Del}|_\Sigma\, S$ is homeomorphic to Σ. A Delaunay refinement algorithm can be driven directly by diagnosing violations of the conditions of the theorem. These violations can be detected by critical point computations and combinatorial checks, which underlie our second surface meshing algorithm.

Neither algorithm is very practical, because both algorithms require expensive computations of either local feature sizes or critical points. Our third surface meshing algorithm combines

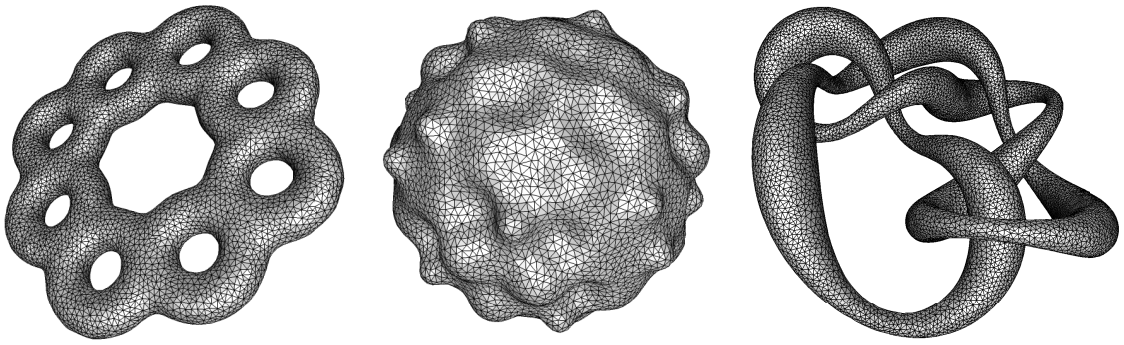


Figure 14.1: Surface meshes generated by D S .

ideas from both algorithms in a more pragmatic way. In this algorithm, a user-specified size field determines the fineness of the mesh and combinatorial tests guarantee that the final mesh is a 2-manifold, but a correct topology is guaranteed only if the size field is sufficiently small. Figure 14.1 depicts three Delaunay surface meshes produced by the algorithm. All three algorithms can refine the mesh to guarantee that no angle is less than 30° or greater than 120°.

The surface meshing algorithms generate a restricted Delaunay triangulation, which is a subcomplex of a Delaunay tetrahedralization, so the volume enclosed by the surface is already meshed by Delaunay tetrahedra. Of course, these tetrahedra have poor shapes unless we refine them by inserting new vertices at their circumcenters, as we do for polyhedral domains. These vertex insertions may delete some surface triangles, so an encroachment rule and several other refinement rules for surface triangles act to maintain domain conformity. We finish the chapter with a guaranteed-quality tetrahedral mesh generation algorithm for volumes bounded by smooth surfaces, which enforces an upper bound on the radius-edge ratios just a little worse than 2.

Throughout this chapter, Σ is a *smooth surface*, our shorthand for a compact (bounded), C^2-smooth 2-manifold without boundary, embedded in $\mathbb{R}^3$. We also assume that Σ is connected. If a surface has multiple connected components, each component can be meshed separately.

14.1 Delaunay surface meshing with a known local feature size

For our first Delaunay refinement surface meshing algorithm, the user supplies a 1-Lipschitz function $\lambda : \Sigma \to \mathbb{R}$, called a *size field*, that reflects the locally desired spacing of vertices on the surface. The algorithm promises to produce no triangle having an empty circumball centered at $c \in \Sigma$ with radius greater than $\lambda(c)$, which implies that no triangle has a circumradius greater than $\lambda(c)$. We require that $\inf_{x\in\Sigma} \lambda(x) > 0$. To guarantee that the algorithm will produce a topologically correct mesh, we also require that $\lambda(x) \leq \varepsilon f(x)$ for some $\varepsilon \in (0, 0.08]$. As we have mentioned, it can be quite difficult to estimate the value of f, but in many practical applications, the user desires elements smaller than those necessary for correctness.

The algorithm refines a restricted Delaunay triangulation by sampling new points on Σ at the centers of circumballs of restricted Delaunay triangles, and adding them to the vertex set. Early in the algorithm's progress, when the sample S is still sparse, $\mathrm{Del}|_\Sigma S$ can contain edges and vertices that are not faces of any triangle. There is no need for the algorithm to keep track of these dangling

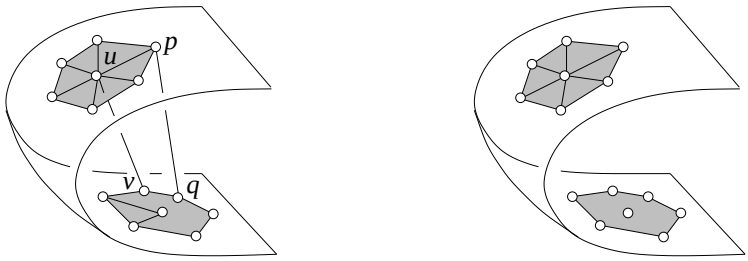


Figure 14.2: The restricted Delaunay edges *pq* and *uv*, which are not faces of any restricted Delaunay triangle, are present in $\mathrm{Del}|_\Sigma S$ (left) but absent by definition from $\mathrm{Del}|^2_\Sigma S$ (right).

simplices, so we focus on the subcomplex of $\mathrm{Del}|_\Sigma S$ consisting of restricted Delaunay triangles and their faces. We call this subcomplex the *restricted Delaunay* 2*-subcomplex*, denoted

$$\mathrm{Del}|^2_\Sigma S = \{\tau : \tau \text{ is a face of a triangle in } \mathrm{Del}|_\Sigma S\}.$$

Recall that a simplex is a face of itself; thus, $\mathrm{Del}|^2_\Sigma S$ contains every triangle in $\mathrm{Del}|_\Sigma S$ with its edges and vertices, but omits dangling edges and vertices, as Figure 14.2 illustrates. Later, we will introduce a *restricted Delaunay* 3*-subcomplex* for tetrahedral mesh generation.

Definition 14.1 (surface Delaunay ball)**.** A *surface Delaunay ball* of a restricted Delaunay triangle $\sigma \in \mathrm{Del}|_\Sigma S$ is a circumball of σ whose center lies both on Σ and on the Voronoi edge σ^* dual to σ.

Every restricted Voronoi vertex is the center of a surface Delaunay ball. Ideally, for every triangle $\sigma \in \mathrm{Del}|_\Sigma S$ there would be exactly one point in $\Sigma \cap \sigma^*$, but for a sparse sample there can be many, so a triangle can have many surface Delaunay balls. A surface Delaunay ball's interior contains no vertex of S because it is centered on σ^*, the Voronoi edge whose generating sites are the vertices of σ.

The Delaunay refinement algorithm splits triangles whose surface Delaunay balls are too large. Specifically, if a surface Delaunay ball $B(c, r)$ has $r > \lambda(c)$, the algorithm inserts its center c into S and updates the Delaunay tetrahedralization $\mathrm{Del}\, S$ and the surface triangulation $\mathrm{Del}|^2_\Sigma S$. Note that there is no need to store two separate triangulations; it is only necessary to mark which triangles in $\mathrm{Del}\, S$ are restricted Delaunay. When no large ball survives, every restricted Voronoi cell $V_p|_\Sigma$ has vertices close to its generating site p; specifically, Σ_p is ε-small for some appropriate ε (recall Definition 13.4). By the Surface Discretization Theorem (Theorem 13.22), the underlying space of $\mathrm{Del}|_\Sigma S$ is homeomorphic to Σ and related by an isotopy, and $\mathrm{Del}|_\Sigma S$ approximates Σ well geometrically. At this time, $\mathrm{Del}|_\Sigma S = \mathrm{Del}|^2_\Sigma S$.

To bootstrap the algorithm, the sample S is initially a set containing the three vertices of a triangle chosen to be small enough that it remains restricted Delaunay throughout execution. We call it the *persistent triangle*. Its presence guarantees that at least one Voronoi edge intersects the surface as required by the Surface Discretization Theorem, namely the persistent triangle's dual edge. Initially, the dual edge is a line that intersects Σ in at least two points, one inducing a small surface Delaunay ball and one inducing a large one, at whose center the fourth vertex is inserted. We summarize the algorithm in pseudocode.

D S 1(Σ, λ)

1. Compute a persistent triangle with vertices on Σ. Let S be the set of vertices of the persistent triangle. Compute Del S and Del$|^2_\Sigma S$.
2. While some triangle in Del$|^2_\Sigma S$ has a surface Delaunay ball $B(c, r)$ with $r > \lambda(c)$, insert c into S, update Del S and Del$|^2_\Sigma S$, and repeat step 2.
3. Return Del$|^2_\Sigma S$.

D S 1 is not concerned with the quality of the triangles it generates, but a small modification can change that by refining skinny triangles as well as oversized ones; see Section 14.4.

A sore point is how to compute a persistent triangle. It is unclear how to do so deterministically; the following randomized heuristic performs well in practice. Pick a point $x \in \Sigma$. Randomly pick three points in $\Sigma \cap B(x, \lambda(x)/7)$ to form a triangle σ. Accept σ as the persistent triangle if $\Sigma \cap B(x, \lambda(x)/7)$ intersects σ's dual line, the line perpendicular to σ through its circumcenter. Otherwise, throw away the vertices of σ and try again. We justify this heuristic in the next section (Proposition 14.3). The only reason we require λ to be 1-Lipschitz is to ensure that σ is persistent.

For D S 1 to run quickly, it should maintain a queue containing all the surface Delaunay balls whose radii exceed the threshold. When a new vertex is inserted into Del S, each new triangular face σ is tested to see if its dual Voronoi edge intersects Σ, and if so, at what point or points. This computation determines both whether σ is restricted Delaunay and whether its surface Delaunay balls are too large; oversized balls are enqueued. Step 2 of the algorithm is a loop that begins by dequeueing a ball and checking whether it is still a surface Delaunay ball. Be forewarned that it is not enough to check whether σ is still in Del$|^2_\Sigma S$; refinement may have shortened σ's dual edge without eliminating it, in which case σ may still be restricted Delaunay but have fewer surface Delaunay balls than it had when it was created.

14.1.1 Proof of termination and guarantees

We first show that D S 1 terminates.

Proposition 14.1. *If* Σ *is compact and* $\inf_{x\in\Sigma} \lambda(x) > 0$, D S 1 *terminates.*

P . D S 1 inserts vertices only at the centers of empty open balls whose radii exceed $\inf_{x\in\Sigma} \lambda(x)$, so no two vertices in S are ever closer to each other than that—except that the vertices of the persistent triangle may be closer to each other. As Σ is bounded, the Packing Lemma (Lemma 6.1) states that there is an upper bound on the number of vertices S can contain. □

The following theorem lists properties of a mesh generated by D S 1.

Theorem 14.2. *Let* $\Sigma \subset \mathbb{R}^3$ *be a* C^2*-smooth, compact, connected surface without boundary. Let* $\lambda : \Sigma \to \mathbb{R}$ *be a 1-Lipschitz function and* $\varepsilon \leq 0.08$ *a value such that for every point* $x \in \Sigma$, $\lambda(x) \leq \varepsilon f(x)$. *Suppose* $\inf_{x\in\Sigma} \lambda(x) > 0$. *Then* D S 1(Σ, λ) *returns a mesh* $\mathcal{T}$ *with the following properties.*

(i) $|\mathcal{T}|$ *can be oriented so that for every triangle* $\sigma \in \mathcal{T}$ *and every vertex* p *of* σ, *the angle between* $\mathbf{n}_p$ *and the oriented normal* $\mathbf{n}_\sigma$ *of* σ *is less than* $7\varepsilon/(1-\varepsilon) < 0.61$ *radians.*

(ii) *Every two distinct triangles in $\mathcal{T}$ that share an edge meet at a dihedral angle greater than $\pi - 14\varepsilon/(1-\varepsilon)$.*

(iii) *The nearest point map $\nu : |\mathcal{T}| \to \Sigma, z \mapsto \tilde{z}$ is a homeomorphism between $|\mathcal{T}|$ and Σ that induces an isotopy.*

(iv) *For every point $z \in |\mathcal{T}|$, $d(z,\tilde{z}) < \frac{15\varepsilon^2}{(1-\varepsilon)^2} f(\tilde{z}) < 0.12 f(\tilde{z})$.*

(v) *The set of vertices in $\mathcal{T}$ is an $\varepsilon/(1-2\varepsilon)$-sample of Σ.*

P . Let S be the set of vertices in $\mathcal{T}$, and recall that $\mathcal{T} = \mathrm{Del}|^2_\Sigma S$. Let p be a vertex in S whose restricted Voronoi cell $V_p|_\Sigma$ has a restricted Voronoi vertex x, which is therefore a vertex of $\Sigma_p \subseteq V_p|_\Sigma$. Upon termination, $d(p,x) \le \lambda(x)$; otherwise, D S 1 would have inserted a vertex at x. Because $\lambda(x) \le \varepsilon f(x)$,

$$d(p,x) \le \varepsilon f(x) \le \frac{\varepsilon}{1-\varepsilon} f(p),$$

the last step following from the Feature Translation Lemma (Lemma 12.2). Therefore, Σ_p is $\varepsilon/(1-\varepsilon)$-small for every $p \in S$. Because $\varepsilon \le 0.08$, $\varepsilon/(1-\varepsilon) \le 0.09$.

By Proposition 14.3 below, the presence of the persistent triangle guarantees that some edge in $\mathrm{Vor}\, S$ intersects Σ. Therefore, the premises of the Surface Discretization Theorem (Theorem 13.22) are satisfied, and $\mathrm{Del}|_\Sigma S$ has properties (i)–(iv). Property (iii) implies that $\mathrm{Del}|_\Sigma S$ has no dangling simplex that is not a face of a triangle, so $\mathrm{Del}|_\Sigma S = \mathrm{Del}|^2_\Sigma S = \mathcal{T}$. Because Σ_p is $\varepsilon/(1-\varepsilon)$-small for every $p \in S$, the Voronoi Intersection Theorem (Theorem 13.14) states that S is an $\varepsilon/(1-2\varepsilon)$-sample of Σ. □

Proposition 14.3. *The triangle σ computed as described at the end of Section 14.1 is persistent—it is present in the output mesh.*

P . By construction, $\Sigma \cap B(x, \lambda(x)/7)$ contains the vertices of σ and intersects the line perpendicular to σ through the circumcenter of σ. Therefore, every surface Delaunay ball of σ centered at a point in $\Sigma \cap B(x, \lambda(x)/7)$ has radius at most $2\lambda(x)/7$, and is included in $B(x, 3\lambda(x)/7)$. We will see that D S 1 never inserts a vertex in $B(x, 3\lambda(x)/7)$. Therefore, there is a surface Delaunay ball of σ that remains empty throughout the algorithm, and σ is a persistent triangle.

Let p be a vertex inserted by D S 1 other than σ's vertices. Assume for the sake of contradiction that $d(p,x) \le 3\lambda(x)/7$. Let v be a vertex of σ. The 1-Lipschitz property of λ implies that $\lambda(p) \ge \lambda(x) - d(p,x) \ge 4\lambda(x)/7 \ge d(p,x) + d(v,x) \ge d(p,v)$.

As D S 1 inserts p at the center of a surface Delaunay ball with radius greater than $\lambda(p)$, the vertex v lies inside this ball, contradicting the fact that every surface Delaunay ball is empty. □

14.1.2 Deleting two vertices of the persistent triangle

The vertices of the persistent triangle are much closer to each other than the user requested. We can fix this by deleting two of them from $\mathrm{Del}\, S$. Unfortunately, their deletion may yield a surface mesh with the wrong topology, or create restricted Delaunay triangles that have surface Delaunay balls a bit larger than requested and a bit too large for the guarantees of Theorem 14.2 to apply.

We can mitigate this problem by choosing the vertices of the persistent triangle to be very close together, but not so close as to cause numerical problems. We can fix the problem entirely by refining the mesh a little bit more after removing the two vertices. Unfortunately, it is possible that this refinement can create a sample set whose Voronoi diagram has no edge that intersects Σ—that is, $\mathrm{Del}|_\Sigma S$ contains no triangle and the algorithm is stuck. At any rate, we do not know how to prove that this never happens, though it seems utterly unlikely in practice. For the sake of theoretical certainty, we discuss here a simple way to delete vertices of the persistent triangle while maintaining the guarantees of Theorem 14.2.

Let v and w be the two vertices of the persistent triangle we wish to delete. When D S 1 terminates, it returns a triangulation of a 2-manifold, so $\mathrm{Del}|_\Sigma S$ contains at least one triangle σ that does not have v for a vertex. The Voronoi edge σ^* dual to σ intersects Σ. Deleting v from S may have the effect of lengthening σ^*, but not of shortening or deleting it, so σ^* still intersects Σ. If the deletion of v causes the mesh to no longer satisfy the antecedents of Theorem 14.2, or if it creates a triangle that is too large, then continue to refine the mesh.

It is possible, as we have said, for the algorithm to get stuck if $\mathrm{Del}|_\Sigma S$ contains no triangles after a vertex insertion. In that unlikely event, fix it by inserting v again, thereby reintroducing the persistent triangle. The key observation is that after v is deleted, it is always possible to insert at least one vertex besides v before getting stuck. Perhaps v will be deleted and reinserted many times, but the algorithm will make progress, and by the Packing Lemma it will terminate.

After producing a satisfactory mesh lacking v, delete w and continue refinement until the mesh is again satisfactory or the algorithm gets stuck. In the latter case, reinsert *both* v and w and continue refining until v can be removed again. Again, we argue that after w is deleted, at least one new vertex can be inserted before v and w must be reintroduced, so even if both vertices are deleted and reinserted many times, the algorithm will make progress and eventually succeed.

14.1.3 Computing edge-surface intersections

The crucial numerical computation in most surface meshing algorithms, including D S 1, is finding the points on a surface that intersect a line or edge. There is no universal algorithm to perform this query, because there are many different ways to represent surfaces, each requiring a different algorithm. A surface meshing program should abstract this computation as a black box subroutine whose replacement allows the mesher to work with different surface representations. Most solid modeling programs include an interface by which an application can ask the solid modeler to perform this query.

For most surface representations, the intersection points are solutions of a system of equations. For a triangulated or polygonal surface, the equations are linear and easily solved. For a parametrized spline, the equations are polynomial, and are usually best solved by iterative methods.

Many applications work with an important class of surfaces called *isosurfaces*, also known as *implicit surfaces*. An isosurface is induced by a smooth function $h : \mathbb{R}^3 \to \mathbb{R}$. For any real number η, the point set $\Sigma = h^{-1}(\eta) = \{p : h(p) = \eta\}$ is an isosurface of h having isovalue η. If η is not a critical value of h, Σ is a smooth surface.

Depending on the nature of h, evaluating an isosurface can be arbitrarily difficult. But if $h(p)$ is polynomial in p, the intersection of a line with an isosurface of h is computed by finding the roots of a univariate polynomial. More commonly, $h(p)$ is piecewise polynomial, and a root

finding computation must be done for each piece that intersects the line or edge. Isosurfaces of piecewise polynomials are rarely C^2-smooth, but Delaunay surface meshing algorithms tend to work well in practice anyway.

Isosurfaces are often created from *voxel data* such as medical images. Voxel data specifies the value of h at each vertex of a cubical grid, but h is not known anywhere else. There are many triangulation algorithms for voxel data, some very fast, but most of them produce some skinny triangles. Algorithms like D S 1 can extract a high-quality mesh of a kidney from a medical image by first extending the domain of h from the vertices to the cubes by interpolation, then meshing an isosurface of the interpolated function. The interpolation basis is usually piecewise polynomial, most commonly piecewise trilinear. Of course, not all discrete point data sets use a cubical grid. For instance, isosurfaces can be generated from irregularly placed data points by constructing their Delaunay triangulation and interpolating the data with piecewise linear functions or more sophisticated methods such as natural neighbor interpolation.

Call a Voronoi edge *bipolar* if the values of h at its two vertices have opposite signs, indicating that one vertex is enclosed by Σ and one is not. If all intersections are transverse, a bipolar edge intersects Σ an odd number of times. If a more sophisticated root finding procedure is not available, a point where a bipolar edge intersects Σ can be approximated by repeated bisection, the secant method, or if h is sufficiently smooth, the Newton–Raphson method.

A good strategy for speed is to delay computing edge-surface intersections until they are needed and to avoid identifying non-bipolar edges that intersect Σ until it becomes necessary. Maintain a queue of Delaunay triangles that dualize to bipolar edges and have not yet been checked. Repeatedly remove a triangle σ from the queue and check whether its dual Voronoi edge σ^* is still bipolar. If so, compute the intersection points in $\sigma^* \cap \Sigma$ and decide if any of the surface Delaunay balls centered at those points is too large. D S 1 often succeeds in practice without considering non-bipolar edges (except when inserting the fourth vertex; there are no bipolar edges when S contains only the vertices of the persistent triangle). Only if the mesh is inadequate when the queue runs empty must the algorithm resort to testing non-bipolar edges for surface intersections.

Figure 14.3 shows meshes of voxel data obtained by using trilinear interpolation to define isosurfaces and a variant of D S 1 described in Section 14.3 to create surface meshes. Instead of computing a persistent triangle, we took intersections of the isosurface with the lines of the voxel grid as seed vertices.

14.2 Topology-driven surface meshing

The algorithm D S 1 can fail if λ does not satisfy the condition $\lambda(x) \leq 0.08 f(x)$ for all $x \in \Sigma$. Unfortunately, it is difficult to estimate $f(x)$, so it is difficult to know whether a user's size field is acceptable. Moreover, the constant 0.08 arising from the theory is conservative; much sparser samples often suffice in practice. It is important to know when the algorithm can stop early.

Our second surface meshing algorithm does not need to know anything about the local feature sizes. For simplicity, we will dispense with the size field λ and suppose that the user desires a topologically correct mesh with no more vertices than the algorithm reasonably requires. Refinement is driven solely by the wish to satisfy the topological ball property (TBP). Of course, the algorithm can optionally incorporate a size field too.

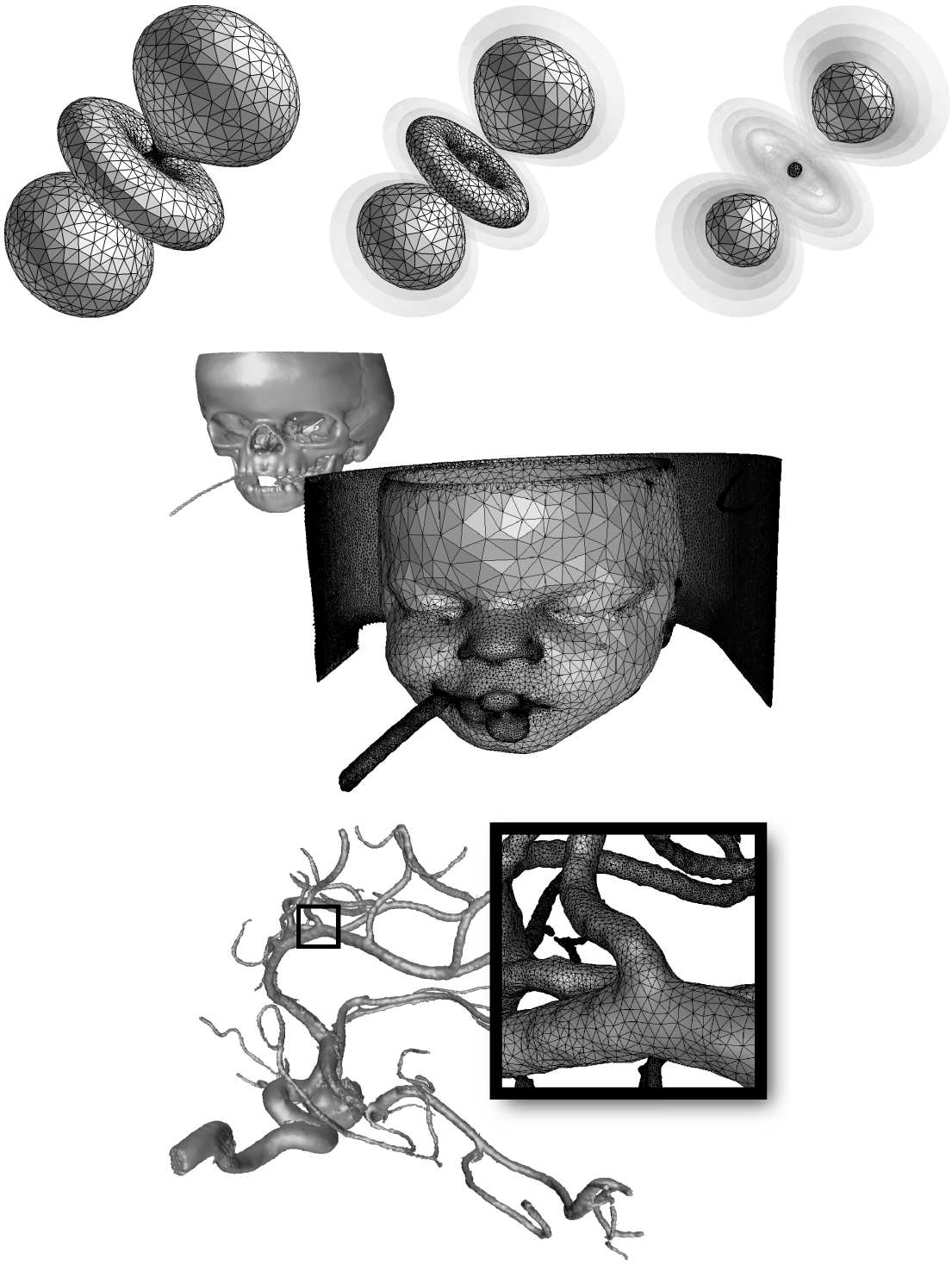


Figure 14.3: Meshes produced by D S of isosurfaces from the A data set. In the top row, three different isovalues provide three different isosurfaces. The bottom two meshes are generated from medical images.

Unfortunately, the algorithm requires that Σ be C^4-smooth, whereas D S 1 requires only C^2-smoothness. A collection of critical point computations is essential to the algorithm, so Σ must have a representation that makes them possible.

Recall from Section 13.2 that the pair (S, Σ) satisfies the TBP if every Voronoi k-face in Vor S that intersects Σ does so transversally in a topological $(k-1)$-ball. In that case, the Topological Ball Theorem (Theorem 13.1) states that the underlying space of $\mathrm{Del}|_\Sigma S$ is homeomorphic to Σ. The topology-driven surface meshing algorithm searches for violations of the TBP and attempts to repair them by sampling points from the surface. It terminates only when (S, Σ) satisfies the TBP. The algorithm maintains a minimum distance between sample points, thereby guaranteeing termination. Before we describe the algorithm, we study the subroutines that diagnose violations of the TBP.

14.2.1 Diagnosing violations of the topological ball property

To find violations of the topological ball property, we define four subroutines V E , T D , V F , and S . When one of them identifies a violation, it samples a new point from Σ for use as a mesh vertex. For simplicity we assume that no Voronoi vertex lies on Σ.

V E checks whether a Voronoi edge e satisfies the TBP. A Voronoi edge should either be disjoint from Σ or intersect Σ transversally at a single point, a 0-ball. Let σ be a restricted Delaunay triangle dual to e, and let $c_{\max}(\sigma) \in e \cap \Sigma$ and $r_{\max}(\sigma)$ be the center and the radius of σ's largest surface Delaunay ball.

V E (e)

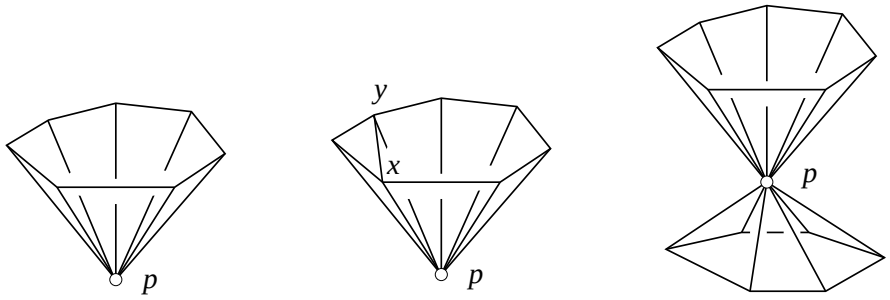
> If e intersects Σ tangentially or at multiple points, let σ be the triangle dual to e and return $c_{\max}(\sigma)$. Otherwise, return null.

T D checks whether the set T_p of restricted Delaunay triangles adjoining a vertex p forms a topological disk. Figure 14.4 illustrates the test: $|T_p|$ is a topological disk if and only if every edge adjoining p in a triangle in T_p is an edge of exactly two triangles in T_p, and there is only one cycle of triangles around p, rather than two or more.

If every Voronoi edge intersects Σ in at most one point (enforced by V E), then T D detects two types of TBP violations. First, it diagnoses a Voronoi facet g that intersects Σ in two or more topological intervals. When this violation happens, the endpoints of the topological intervals each lie on a different edge of g (thanks to V E), so there are at least four such edges. Their dual Delaunay triangles all share the edge dual to g, so the dual edge is not an edge of exactly two restricted Delaunay triangles. Second, T D diagnoses a Voronoi cell V_p whose boundary intersects Σ in two or more loops that meet Voronoi edges. This violation implies that T_p is a union of two or more topological disks joined at p, as in Figure 14.4(c).

T D (p)

1. Let T_p be the set of triangles in $\mathrm{Del}|^2_\Sigma S$ that adjoin p.
2. If $|T_p|$ is empty or a topological disk, return null.
3. Let σ be the triangle that maximizes $r_{\max}(\sigma)$ among T_p. Return $c_{\max}(\sigma)$.



(a) (b) (c)

Figure 14.4: A test for whether the triangles adjoining p form a topological disk. (a) $|T_p|$ is a topological disk. (b) The edges px and py are not edges of exactly two triangles, so $|T_p|$ is not a topological disk. (c) There is more than one cycle of triangles, so $|T_p|$ is a union of topological disks joined at p.

If a Voronoi facet intersects Σ, the intersection should be transverse and it should be a single topological interval. This property is violated if (i) the Voronoi facet intersects Σ tangentially, (ii) the Voronoi facet intersects Σ in two or more topological intervals, or (iii) the Voronoi facet intersects Σ in a loop. All three violations (i), (ii), and (iii) can happen simultaneously.

As we have discussed, violation (ii) is detected by T D . The next subroutine, V F , detects violations (i) and (iii). V F uses critical point computations. Let C be a smooth loop on a plane. Given a direction d parallel to the plane, the critical points of C in direction d are the points where d is normal to C.

V F (g)

1. If there exists a point in $g \cap \Sigma$ where g and Σ have the same tangent plane, return that point.
2. Let $\Pi = \text{aff}\, g$. Choose a random vector d parallel to Π. Compute the set X of critical points of the curves in $\Pi \cap \Sigma$ in the direction d.
3. If no point in X lies on g, return null.
4. As g intersects Σ transversally, $g \cap \Sigma$ is a collection of disjoint simple curves (intervals or loops) and $X \cap g$ is the set of critical points of these curves in direction d. Let V_p be a Voronoi cell with face g. For each $x \in X \cap g$,
   (a) Compute the line $\ell_x \subset \Pi$ through x parallel to d. The line ℓ_x is normal to $g \cap \Sigma$ at x.
   (b) Compute $Y \leftarrow \ell_x \cap g \cap \Sigma$. If Y contains two or more points, return the point in Y furthest from the vertex p dual to V_p.
5. Return null.

Step 2 of V F assumes that $\Pi \cap \Sigma$ has finitely many critical points in a direction. It is known that the critical points in any direction are isolated if Σ is C^3-smooth, and the number of critical points is finite for algebraic surfaces of fixed degree.

The three subroutines V E , T D , and V F ensure that the Voronoi edges and Voronoi facets in Vor S satisfy the TBP. The TBP has one other requirement: that each Voronoi cell intersect Σ in a topological disk or not at all. The subroutines T D and V F ensure that for each Voronoi cell V_p, $(\mathrm{Bd}\, V_p) \cap \Sigma$ is either empty or a single loop that crosses more than one facet of V_p. Hence, the only ways that V_p can violate the TBP are if $V_p \cap \Sigma = \Sigma$ (we assume Σ is connected) or $V_p \cap \Sigma$ has a handle. We diagnose these cases by checking the *silhouettes* of Σ.

Definition 14.2 (silhouette)**.** For a smooth surface Σ and a specified direction d, the silhouette J_d is the set of points $\{x \in \Sigma : \mathbf{n}_x \cdot d = 0\}$. That is, the normal to Σ at each point in J_d is orthogonal to the direction d.

The following result motivates the use of silhouettes.

Proposition 14.4. *Let δ be a connected component of $V_p \cap \Sigma$ for some Voronoi site p. If* $\mathrm{Bd}\,\delta$ *is empty, then for every direction d, $\delta \cap J_d \neq \emptyset$. If* $\mathrm{Bd}\,\delta$ *is a single loop and $\delta \cap J_d = \emptyset$ for some direction d, then δ is a topological disk.*

P . If $\mathrm{Bd}\,\delta$ is empty, then δ is the surface Σ and so $\delta \cap J_d = J_d$. It is clear in this case that J_d is nonempty for every direction d. Suppose that $\mathrm{Bd}\,\delta$ is a single loop. Let H be a plane perpendicular to d. Consider the map $\varphi : \delta \to H$ that projects each point of δ orthogonally to H. Since δ is connected and compact, it suffices to prove that φ is injective.

Assume to the contrary that φ is not injective. Then there is a line ℓ parallel to d that intersects δ in two or more points. Let x and y be two consecutive intersection points along ℓ.

As $\delta \cap J_d = \emptyset$, neither x nor y belongs to J_d, which means that neither $\mathbf{n}_x$ nor $\mathbf{n}_y$ is orthogonal to d. Because x and y are consecutive in $\ell \cap \delta$, $\mathbf{n}_x$ and $\mathbf{n}_y$ are oppositely oriented in the sense that the inner products $\mathbf{n}_x \cdot d$ and $\mathbf{n}_y \cdot d$ have opposite signs. Because δ is connected, there is a smooth curve $\rho \subset \delta$ connecting x and y. The normal to Σ changes smoothly from $\mathbf{n}_x$ to $\mathbf{n}_y$ along ρ. By the mean value theorem, there is a point $z \in \rho$ such that $\mathbf{n}_z$ is orthogonal to d. But then $z \in \delta \cap J_d$, contradicting the emptiness of $\delta \cap J_d$. □

The subroutine S takes advantage of Proposition 14.4. It checks if a Voronoi cell V_p intersects the silhouette J_d, where $d = \mathbf{n}_p$ is normal to Σ at p. If $V_p \cap J_d \neq \emptyset$, either J_d intersects some facets of V_p, or V_p contains a component of J_d. The first possibility is checked by a subroutine S P $(\Sigma, \Pi, \mathbf{n}_p)$ that returns the points where J_d intersects the affine hull Π of a facet of V_p. The second possibility can be detected by checking if V_p contains a critical point of J_d in a direction d' orthogonal to d. To perform this check, we require that J_d be a set of smooth loops, and that the number of critical points on J_d along a direction be finite. The first requirement is likely true if d is obtained by applying a tiny, random perturbation to $\mathbf{n}_p$; for simplicity, assume it holds. The second requirement holds if Σ is C^4-smooth. Let S C (Σ, d, d') be the subroutine that returns the critical points of J_d in the direction d' (see Exercise 5).

S (p)

1. Let d be $\mathbf{n}_p$ after a tiny random perturbation.
2. Choose a random direction d' orthogonal to d.
3. Compute $X \leftarrow$ S C (Σ, d, d').

4. If X contains a point inside V_p, return it.
5. Otherwise, for each facet g of V_p:
   (a) Compute $Y \leftarrow$ S P (Σ, Π, d) where $\Pi = \text{aff}\, g$.
   (b) If Y contains a point on g, return it.
6. Return null.

The following proposition gives a lower bound on the distance between a site p and every silhouette point in V_p.

Proposition 14.5. *Let p be a Voronoi site. For every point $x \in V_p \cap J_{\mathbf{n}_p}$, $d(p, x) \geq f(p)/3$.*

P . If $d(p, x) < f(p)/3$, the Normal Variation Theorem (Theorem 12.8) implies that $\angle(\mathbf{n}_p, \mathbf{n}_x) \leq \alpha(1/3)$, contradicting the fact that $\mathbf{n}_x$ is orthogonal to $\mathbf{n}_p$ by definition. □

14.2.2 A topology-driven Delaunay refinement algorithm

The topology-driven Delaunay refinement algorithm initializes the sample S with a set of points called *seeds*, which are critical points of Σ in a chosen direction. Then it repeatedly calls the subroutines of Section 14.2.1 to search for violations of the topological ball property. When a violation is found, a new vertex is added to S. When no violation survives, the underlying space of $\text{Del}|_\Sigma S$ is homeomorphic to Σ, and the algorithm returns $\text{Del}|^2_\Sigma S$, which must be equal to $\text{Del}|_\Sigma S$. It is possible that some seeds are too close together, so that the surface triangulation may have unnecessarily short edges adjoining the seeds. This problem can be fixed by deleting the seeds, then refining again to restore the correct topology.

D S 2(Σ)

1. Let S be a set of seed points on Σ. Compute $\text{Vor}\, S$ and $\text{Del}|^2_\Sigma S$.
2. Perform steps (a)–(d) below in order. As soon as a non-null point x is returned, terminate the current loop, skip the remaining steps, and go to step 3.
   (a) For every Voronoi edge e in $\text{Vor}\, S$, compute $x \leftarrow$ V E (e).
   (b) For every $p \in S$, compute $x \leftarrow$ T D (p).
   (c) For every Voronoi facet g in $\text{Vor}\, S$, compute $x \leftarrow$ V F (g).
   (d) For every $p \in S$, compute $x \leftarrow$ S (p).
3. If x is non-null, insert x into S, update $\text{Vor}\, S$ and $\text{Del}|^2_\Sigma S$, and go to step 2.
4. Otherwise, return $\text{Del}|^2_\Sigma S$.

Instead of computing the seed set, D S 2 could begin with just a single sample point in S; S would generate more sites on each component of Σ and the algorithm would succeed. However, the computation of critical points of a silhouette in a specified direction is more expensive than computing critical points of Σ.

14.2.3 Proof of termination and homeomorphism

To show that D S 2 terminates, we use the Small Intersection Theorem (Theorem 13.6) to guarantee a lower bound on the distances among the sites inserted into S, excepting distances between pairs of seed points.

Proposition 14.6. D S 2 *terminates.*

P . Let $f_{\min} = \inf_{x \in \Sigma} f(x)$. We claim that every point returned by a subroutine called by D - S 2 is at a distance of at least $0.09 f_{\min}$ from every site in S, so the Packing Lemma (Lemma 6.1) implies that D S 2 terminates.

If V E (e) returns a point, the Voronoi edge e intersects Σ either tangentially or at more than one point. Let p be a site whose Voronoi cell V_p has the edge e. By the Voronoi Edge Lemma (Lemma 13.7), the distance from p to the furthest point in $e \cap \Sigma$ is greater than $0.15 f(p)$. V E returns that furthest point; call it x. The claim follows because no site is closer to x than p.

If T D (p) returns a point, T_p is nonempty and not a topological disk. So Σ_p is nonempty. Σ_p is not 0.09-small; if it were, the Small Intersection Theorem (Theorem 13.6) would imply that $|T_p|$ is a topological disk. It follows that some triangle in T_p has a surface Delaunay ball with radius greater than $0.09 f(p)$. The claim follows because T D returns the center of the largest surface Delaunay ball.

If V F (g) returns a point, let $p \in S$ be a site whose Voronoi cell V_p has the face g. If g intersects Σ tangentially, V F returns a tangential contact point x such that $\mathbf{n}_x = \pm \mathbf{n}_g$, and the contrapositive of Proposition 13.3(ii) implies that $d(x, p) \geq f(p)/3$, thus $d(x, S) \geq f(p)/3$. If g intersects Σ transversally, then V F returns a point x in the mutual intersection of g, Σ, and a line ℓ. There must be at least two such intersection points for V F to return a point. Let y be the intersection point where ℓ is perpendicular to the silhouette $\Sigma \cap g$, and let z be another intersection point. Proposition 13.3(ii) implies that $\angle(yz, \mathbf{n}_y) = \pi/2 - \angle_a(\mathbf{n}_g, \mathbf{n}_y) \leq \alpha(0.09) + \arcsin 0.09 < \alpha(0.09) + \beta(0.09)$. Proposition 13.5 implies that $d(p, z) > 0.09 f(p)$. As V F returns the intersection point furthest from p, the claim follows.

By Proposition 14.5, S (p) returns a point at a distance of $f(p)/3$ or more from p. □

D S 2 inserts sites as long as the TBP is violated, so (S, Σ) satisfies the TBP when D - S 2 terminates. We appeal to the Homeomorphism Theorem (Theorem 13.16).

Theorem 14.7. *Let $\Sigma \subset \mathbb{R}^3$ be a C^4-smooth, compact, connected surface without boundary.* D S 2(Σ) *returns a restricted Delaunay triangulation whose underlying space is homeomorphic to Σ.*

This theorem does not guarantee that the Hausdorff distance between Σ and the output mesh is small—a guarantee that cannot be made unless feature sizes are computed. However, upper bounds on the triangle sizes and lower bounds on the dihedral angles at which triangles meet can be enforced by simply refining triangles that do not meet the bounds; see Section 14.4.

14.3 A practical surface meshing algorithm

Both D S 1 and D S 2 employ computationally expensive predicates. For D S 1, it is rarely practical to implement an oracle that computes local feature sizes, or even a decent lower

bound on them. Computing the exact medial axis of a specified implicit surface is known to be hard. The medial axis can be approximated from a dense sample of the surface, but that requires a dense sample to be available in the first place.

For D S 2, the predicates that compute critical points are computationally expensive and difficult to perform stably. On the bright side, the topological disk test T D performs easy combinatorial checks, but it alone does not suffice to guarantee the homeomorphism of the input surface and the output mesh. Here we discuss a practical compromise between the two algorithms.

Our third surface meshing algorithm uses two tests to drive refinement, one geometric and one topological: the user-supplied size field $\lambda : \Sigma \to \mathbb{R}$ from D S 1 and the test T D from D S 2. A triangle is refined if its surface Delaunay ball is excessively large, or if the triangle is the largest among a group of triangles that adjoin a vertex in a neighborhood that is not a topological disk. If $\lambda(x) \leq \varepsilon f(x)$ for some $\varepsilon \in (0, 0.08]$, the algorithm offers the same guarantees as D S 1, enumerated in Theorem 14.2. Otherwise, the output mesh might not have the same topology as the input surface, but it is guaranteed to be a 2-manifold. In practice, this usually suffices to achieve homeomorphism as well.

D S (Σ, λ)

1. Compute a persistent triangle with vertices on Σ. Let S be the set of vertices of the persistent triangle. Compute $\mathrm{Del}\, S$ and $\mathrm{Del}|^2_\Sigma S$.
2. While some triangle in $\mathrm{Del}|^2_\Sigma S$ has a surface Delaunay ball $B(c, r)$ with $r > \lambda(c)$, insert c into S, update $\mathrm{Del}\, S$ and $\mathrm{Del}|^2_\Sigma S$, and repeat step 2.
3. For every $p \in S$, compute $c \leftarrow$ T D (p); if some c is non-null, stop looping, insert c into S, update $\mathrm{Del}\, S$ and $\mathrm{Del}|^2_\Sigma S$, and go to step 2.
4. If S contains more than one vertex of the persistent triangle, delete one, update $\mathrm{Del}\, S$ and $\mathrm{Del}|^2_\Sigma S$, and go to step 2.
5. Return $\mathrm{Del}|^2_\Sigma S$.

Figure 14.5 depicts meshes generated by D S for three different constant values of λ. As λ decreases, the mesh better captures both the geometry and the topology of the surface.

The next proposition shows that, like Ruppert's algorithm, D S generates a mesh whose size is proportional to the integral over the domain of the inverse squared local feature size—or the inverse squared size field, if the latter is more demanding. There is an asymptotically matching lower bound on the number of vertices needed to respect the size field, so in this sense D S and D S 1 produce size-optimal meshes. Observe that if the size field is constant, this integral is proportional to the area of the surface. In the following bound, we assume that two vertices of the persistent triangle have been deleted, as discussed in Section 14.1.2.

Proposition 14.8. *If λ is 1-Lipschitz and for every point $x \in \Sigma$, $\lambda(x) \leq 0.08 f(x)$, then the mesh generated by* D S (Σ, λ) *has fewer than* $8 \cdot \int_\Sigma dx/\lambda(x)^2$ *vertices.*

P . Every vertex c that step 2 of D S inserts into S is at a distance of at least $\lambda(c)$ from every previously inserted vertex. In step 3, if T D (p) returns a point c, then Σ_p is not 0.09-small; if it were, the Small Intersection Theorem (Theorem 13.6) implies that $|T_p|$ would be a topological disk. Therefore, $d(c, p) \geq 0.09 f(p)$, and as f is 1-Lipschitz, $d(c, p) \geq 0.08 f(c)$.

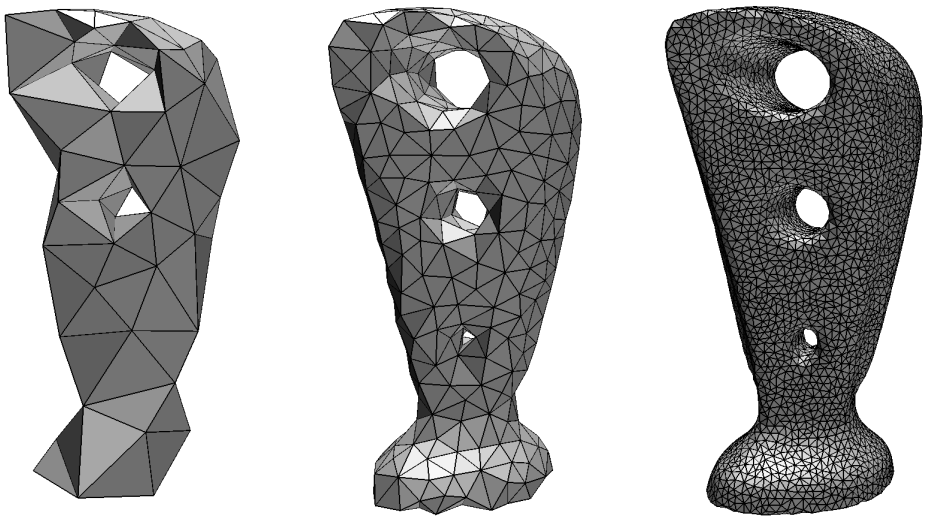


Figure 14.5: Both geometric and topological fidelity improve as the size field decreases.

Because c is inserted in p's Voronoi cell, c is a distance of at least $0.08f(c)$ from every previously inserted vertex. Therefore, every vertex c that D S inserts is a distance of at least $\min\{\lambda(c), 0.08f(c)\} = \lambda(c)$ from every previous vertex. Because λ is 1-Lipschitz, every vertex $p \in S$ is a distance of at least $\lambda(p)/2$ from every other vertex in S.

If we center balls of radii $\lambda(p)/4$ at every vertex $p \in S$, their interiors are pairwise disjoint. Let $\Gamma_p = \Sigma \cap B(p, \lambda(p)/4)$, and let a_p be the area of Γ_p. Because λ is 1-Lipschitz, $\lambda(x) \leq 5\lambda(p)/4$ for every $x \in \Gamma_p$. Thus

$$\begin{aligned} \int_\Sigma \frac{1}{\lambda(x)^2}\,dx &> \sum_{p\in S} \int_{\Gamma_p} \frac{1}{\lambda(x)^2}\,dx \\ &> \sum_{p\in S} \frac{16}{25\lambda(p)^2} a_p \\ \Rightarrow \qquad |S| &< \frac{25}{16}\left(\min_{p\in S} \frac{a_p}{\lambda(p)^2}\right)^{-1} \int_\Sigma \frac{1}{\lambda(x)^2}. \end{aligned}$$

The remainder of the proof shows that $a_p/\lambda(p)^2 \geq 0.196$, from which the proposition follows.

Let B_1 and B_2 be the two medial balls sandwiching the surface Σ at p, with B_1 not larger than B_2. Let D be the geometric disk bounded by the circle at the intersection of the boundaries of $B(p, \lambda(p)/4)$ and B_1, as illustrated in Figure 14.6. Consider the orthogonal projections $\widetilde{D}$ and $\widetilde{\Gamma}_p$ of D and Γ_p, respectively, onto the plane tangent to Σ at p. Because Σ has no boundary, $\widetilde{D}$ is included in $\widetilde{\Gamma}_p$. Therefore,

$$a_p = \text{area}(\Gamma_p) \geq \text{area}(\widetilde{\Gamma}_p) \geq \text{area}(\widetilde{D}) = \text{area}(D). \tag{14.1}$$

We derive a lower bound on the radius of D to bound a_p from below. From the similar triangles

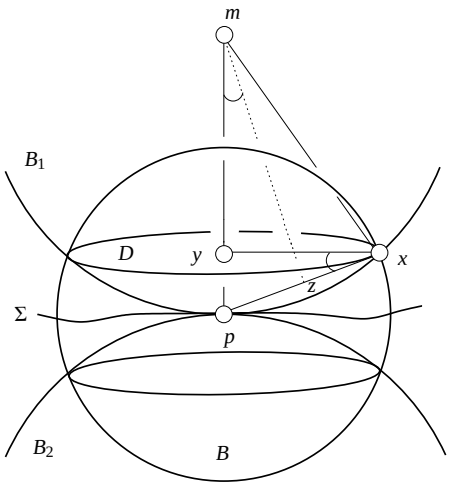


Figure 14.6: Two medial balls B_1 and B_2 sandwich the surface Σ. The projection of D onto the plane tangent to Σ at p is included in the projection of $\Sigma \cap B$ onto the same plane.

pmz and *pxy*,

$$\begin{aligned} \frac{d(p,y)}{d(p,x)} &= \frac{d(p,z)}{d(p,m)} = \frac{d(p,x)}{2\,d(p,m)} \\ \Rightarrow \quad d(p,y) &= \frac{d(p,x)^2}{2\,d(p,m)} = \frac{\lambda(p)^2}{32\,d(p,m)}. \end{aligned}$$

Therefore,

$$\begin{aligned} \text{radius}(D)^2 &= d(p,x)^2 - d(p,y)^2 \\ &= \frac{\lambda(p)^2}{16} - \frac{\lambda(p)^4}{1{,}024\,\text{radius}(B_1)^2}. \end{aligned}$$

As $\text{radius}(B_1) \geq f(p)$, which is greater than $12\,\lambda(p)$ by our assumption that $\lambda(p) \leq 0.08\,f(p)$,

$$\text{radius}(D)^2 \geq \frac{\lambda(p)^2}{16} - \frac{\lambda(p)^2}{147{,}456} > 0.0624\,\lambda(p)^2.$$

Hence, $\text{area}(D)$ is greater than $0.0624\pi\,\lambda(p)^2$, implying that $a_p > 0.196\,\lambda(p)^2$ by (14.1). □

We summarize the properties of meshes generated by D S .

Theorem 14.9. *Let $\Sigma \subset \mathbb{R}^3$ be a C^2-smooth, compact, connected surface without boundary. Suppose* $\inf_{x\in\Sigma} \lambda(x) > 0$. *Then* D S (Σ, λ) *returns a mesh $\mathcal{T}$ with the following properties.*

(i) *The underlying space of $\mathcal{T}$ is a 2-manifold.*

(ii) *If there is a value $\varepsilon \leq 0.08$ such that $\lambda(x) \leq \varepsilon f(x)$ for every point $x \in \Sigma$, then the nearest point map $\nu : |\mathcal{T}| \to \Sigma, z \mapsto \tilde{z}$ is a homeomorphism between $|\mathcal{T}|$ and Σ that induces an isotopy that moves each point $z \in |\mathcal{T}|$ by a distance $d(z,\tilde{z}) < \frac{15\varepsilon^2}{(1-\varepsilon)^2} f(\tilde{z}) < 0.12\,f(\tilde{z})$.*

(iii) *If* $\lambda(x) \leq 0.08 f(x)$ *for every point* $x \in \Sigma$ *and* λ *is* 1*-Lipschitz, then there are fewer than* $8 \cdot \int_\Sigma dx/\lambda(x)^2$ *vertices in* $\mathcal{T}$*. This number is asymptotically optimal: every sample* $S \subset \Sigma$ *such that* $d(x, S) \leq \lambda(x)$ *for every* $x \in \Sigma$ *has* $\Omega\left(\int_\Sigma dx/\lambda(x)^2\right)$ *vertices. In this sense,* $\mathcal{T}$ *is size-optimal.*

P . When D S terminates, every vertex in $\mathcal{T}$ adjoins a set of triangles whose underlying space is a topological disk. By a well-known result of piecewise linear topology, $|\mathcal{T}|$ is a 2-manifold, proving (i). The proof of (ii) is the same as that of Theorem 14.2(iii, iv). The first part of (iii) is Proposition 14.8. The second part of (iii) is Exercise 9 in Chapter 13. □

14.4 Extensions: quality, smoothness, and polyhedral surfaces

It is easy to incorporate refinement steps into the three surface meshing algorithms we have studied to ensure that the triangles of the output mesh have good quality or adjoin each other at nearly-flat dihedral angles, thereby better capturing the geometry of Σ.

Let $\rho(\sigma)$ be the radius-edge ratio of a triangle σ, and let $\bar{\rho}$ be an upper bound on the acceptable radius-edge ratio. Equivalently, $\arcsin 1/(2\bar{\rho})$ is a lower bound on the smallest acceptable angle. The radius-edge ratio threshold can be as small as 1, in which case no triangle in the final mesh has an angle less than $30^\circ = \pi/6$ radians nor an angle greater than $120^\circ = 2\pi/3$ radians. If there is a skinny triangle σ in $\mathrm{Del}|_\Sigma S$, the subroutine Q inserts $c_{\max}(\sigma)$, the center of σ's largest surface Delaunay ball.

Q $(S, \bar{\rho})$

If there is a restricted Delaunay triangle σ in $\mathrm{Del}|^2_\Sigma S$ for which $\rho(\sigma) > \bar{\rho}$, then insert $c_{\max}(\sigma)$ into S, update $\mathrm{Del}\, S$, and update $\mathrm{Del}|^2_\Sigma S$.

As Σ is a smooth surface, the triangles of the mesh adjoin each other at dihedral angles that approach π as the edge lengths approach zero, as Theorem 14.2(ii) shows. The *roughness* of an edge e in $\mathrm{Del}|_\Sigma S$, denoted $\mathrm{rough}(e)$, is π radians minus the dihedral angle at e. If the roughness of an edge exceeds a threshold $\bar{r}$, the subroutine S refines one of the two adjoining triangles. S thereby enforces the restriction that the dihedral angles are at least $\pi - \bar{r}$ radians.

S $(S, \bar{\rho})$

If there is an edge e in $\mathrm{Del}|^2_\Sigma S$ such that $\mathrm{rough}(e) > \bar{r}$, let σ be the restricted Delaunay triangle with the largest surface Delaunay ball among the two triangles having e for an edge, insert $c_{\max}(\sigma)$, update $\mathrm{Del}\, S$, and update $\mathrm{Del}|^2_\Sigma S$.

In the algorithm D S , the calls to Q and S can be added just after step 4. Whenever either call inserts a new vertex, control should return to step 2 to ensure that the mesh respects the size field and is topologically valid. The persistent triangle must be removed before Q is called, or its short edges may cause the region around it to be overrefined.

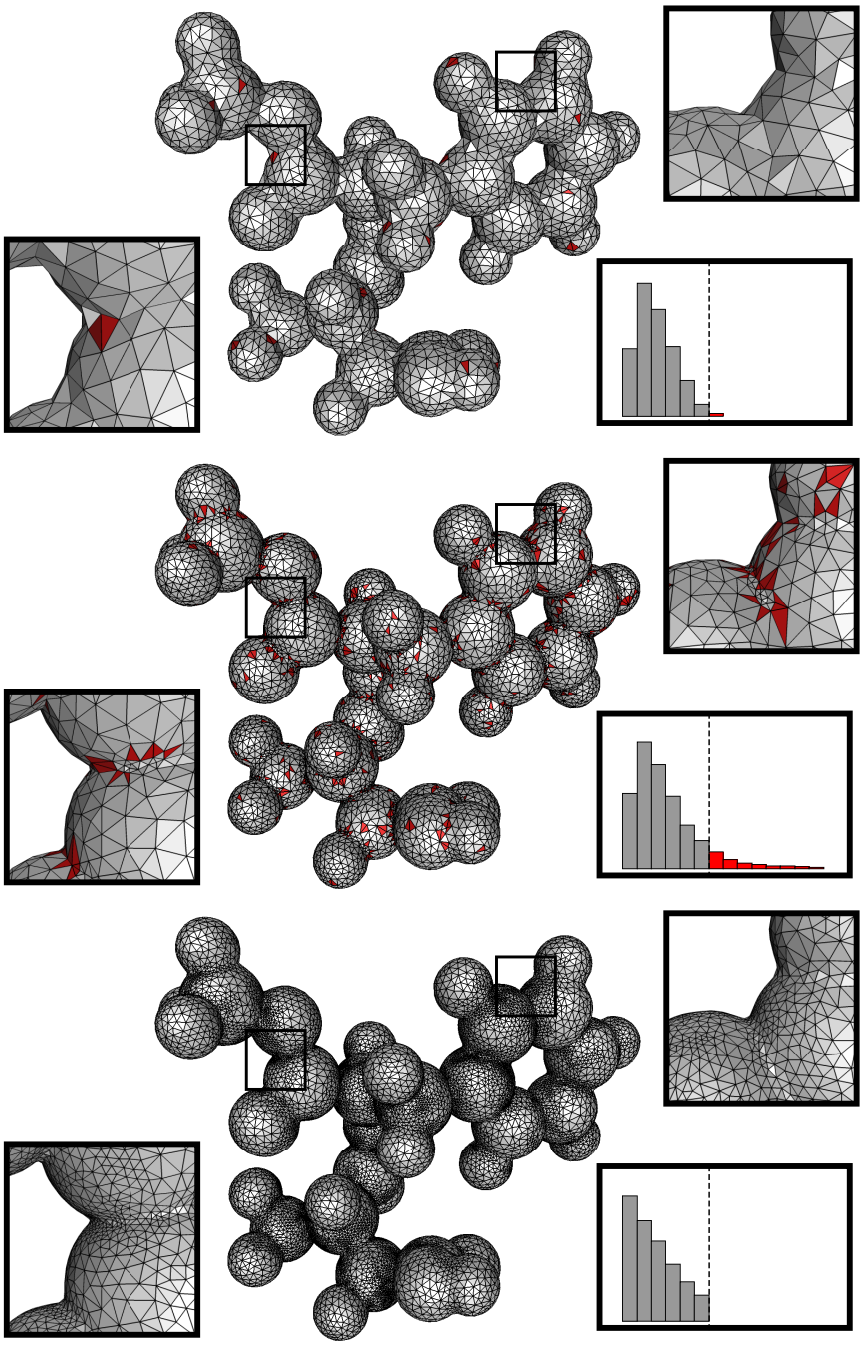


Figure 14.7: The topmost mesh is generated by D S with no attention to triangle quality or the dihedral angles at which adjoining triangles meet. The histogram tabulates the triangles' radius-edge ratios, with only a few triangles (at right) exceeding the threshold. The center mesh shows that refinement intended to improve the dihedral angles succeeds at the expense of triangle quality. The bottom mesh shows the mesh after further refinement to eliminate all the poor-quality triangles.

Figure 14.7 illustrates the effects of S and Q . The mesh at center shows that the quality of the triangles can suffer if we use S alone. The quality can be corrected by using Q as well.

When Q $(S, \bar{\rho})$ splits a skinny triangle with $\bar{\rho} \geq 1$, the newly inserted vertex is at least as far from every other vertex in S as the vertices of the skinny triangle's shortest edge are from each other. Therefore, Q never creates a shorter edge in Del S than the shortest existing edge, so it cannot cause D S to run forever. Neither can S , because for any $\bar{r} > 0$, Theorem 14.2(ii) guarantees that every edge will satisfy the roughness bound once sufficient refinement has taken place.

An alternative way to achieve high quality is to choose a size field λ that does not change too quickly. For example, if λ is 0.5-Lipschitz, no triangle of the final mesh has a radius-edge ratio exceeding 2, nor therefore an angle less than 14.47°. A constant size field guarantees no angle less than 30°, at the cost of losing the flexibility to create a graded mesh. We will use this observation to analyze the grading of the Q subroutine when the size field is not so well-behaved.

Proposition 14.10. *If the size field λ is δ-Lipschitz for $\delta < 1$, and $\lambda(x) \leq 0.08 f(x)$ for every point $x \in \Sigma$,* D S (Σ, λ) *returns a mesh whose triangles have radius-edge ratios at most $1/(1-\delta)$.*

P . Because $\lambda(x) \leq 0.08 f(x)$, D S refines no triangle solely for topological reasons; every vertex $p \in S$ has no vertex within a distance of $\lambda(p)$ at the time p is inserted.

Let σ be an output triangle whose shortest edge has length ℓ and whose surface Delaunay ball has center c and radius r. The circumradius of σ is less than or equal to r. Let p be the most recently inserted vertex of the shortest edge of σ; because it was inserted after the other vertex, $\ell > \lambda(p)$. Because no vertex was inserted at c and λ is δ-Lipschitz, $r \leq \lambda(c) \leq \lambda(p) + \delta r$; rearranging terms gives $r \leq \lambda(p)/(1-\delta)$. Thus $\rho(\sigma) \leq r/\ell \leq 1/(1-\delta)$. □

We use this proposition to analyze Q $(S, \bar{\rho})$, particularly how the grading of the mesh depends on $\bar{\rho}$, when the size field is arbitrary. Define the $(\bar{\rho}-1)/\bar{\rho}$-Lipschitz *regularized size field*

$$\mu(x) = \inf_{y \in \Sigma} \left(\min\{\lambda(y), 0.08 f(y)\} + \frac{\bar{\rho}-1}{\bar{\rho}} d(x,y) \right),$$

and observe that $\mu(x) \leq \lambda(x)$ and $\mu(x) \leq 0.08 f(x)$. We will see that the Lipschitz property captures how refining for quality can affect the edge lengths in the mesh. The more $\bar{\rho}$ exceeds 1, the more strongly the mesh can be graded.

The idea is that if we call D S (Σ, μ) with the regularized size field μ but no Q option, it splits every triangle that would be split by D S (Σ, λ) with the original size field λ and the Q $(S, \bar{\rho})$ option, and possibly other triangles as well. Therefore, the latter does not refine more than the former.

Theorem 14.11. *If $\bar{\rho} \geq 1$,* D S (Σ, λ) *with the* Q $(S, \bar{\rho})$ *option terminates and returns a mesh in which every vertex p is at a distance greater than $\mu(p) \cdot \bar{\rho}/(2\bar{\rho}-1)$ from the nearest distinct vertex.*

P . First, we claim that every vertex c is at a distance greater than $\mu(c)$ from the nearest vertex inserted before it. Suppose for the sake of contradiction that a vertex c is inserted such that $d(c,p) \leq \mu(c)$, where p was inserted prior to c. Suppose that c is the first such vertex. As

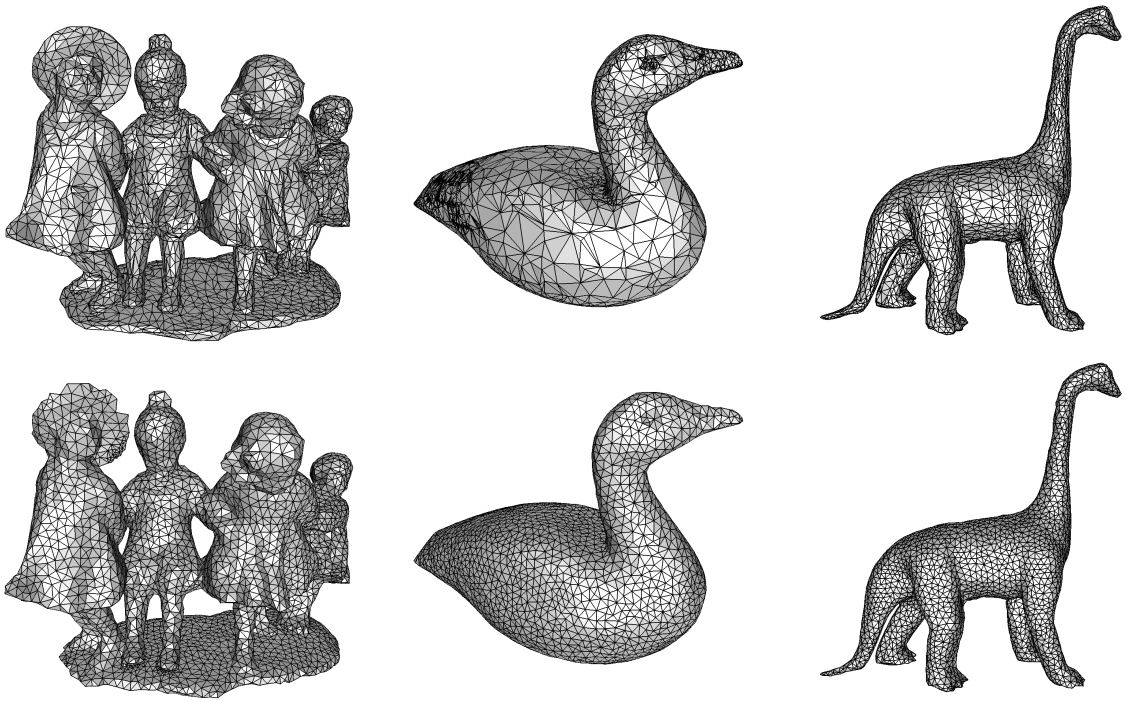


Figure 14.8: Polygonal surfaces (top row) remeshed by D S (bottom row).

$\mu(c) \leq \min\{\lambda(c), 0.08\, f(c)\}$, c must have been inserted by the subroutine Q at the center of a surface Delaunay ball of a skinny triangle.

If D S had been called with the regularized size field $\mu(p)$ but without the Q option, it could have created the same skinny triangle, because c is the first vertex whose insertion is not justified by μ, but D S would have declined to split the skinny triangle. But this contradicts Proposition 14.10. Therefore, $d(c, p) > \mu(c)$.

Consider the same distance from the point of view of a point p inserted before c. By the Lipschitz property of μ, $d(c, p) > \mu(c) \geq \mu(p) - d(c, p) \cdot (\bar{\rho} - 1)/\bar{\rho}$. Rearranging terms gives $d(c, p) > \mu(p) \cdot \bar{\rho}/(2\bar{\rho} - 1)$. □

Although Q $(S, \bar{\rho})$ and a $(\bar{\rho} - 1)/\bar{\rho}$-Lipschitz size field appear to be two theoretically equivalent paths to quality, the former is lazier—it refines a triangle only when really necessary to improve its quality—so it tends to produce meshes with fewer triangles than the theory suggests. It also has the advantages of working well in conjunction with T D and S and not requiring the user to implement a Lipschitz size field.

A *polygonal surface* is a PLC whose underlying space is a 2-manifold without boundary. Although D S is designed to mesh smooth surfaces, it often succeeds in meshing or remeshing polygonal surfaces that closely approximate smooth surfaces, in the sense that their faces meet at dihedral angles close to π. *Remeshing* is the act of replacing an existing mesh with a new one to improve its quality, to make it finer or coarser, or to make its faces be Delaunay, as illustrated in Figure 14.8. The initial Delaunay triangulation can be built from the vertices of the input surface,

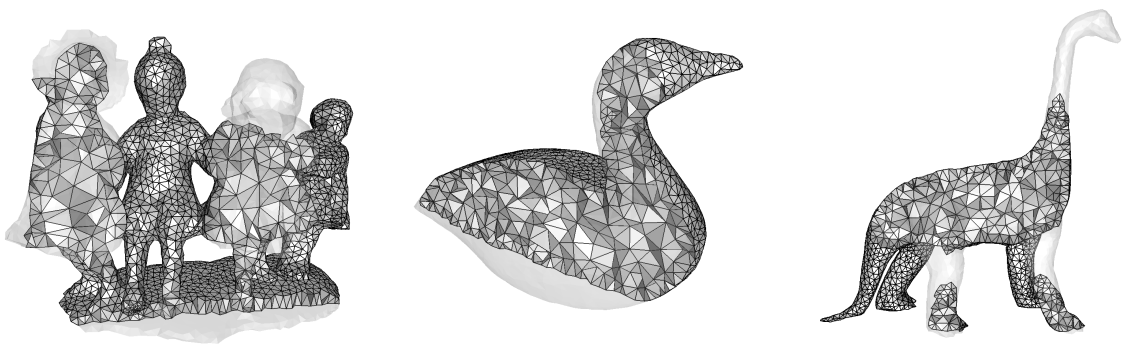


Figure 14.9: Tetrahedral meshes generated by D T S .

but if the goal is to produce a coarser mesh, or if the input surface has closely spaced vertices that are harmful to mesh quality, a well-spaced subset of the input vertices is a better start. Note that in practice, we have never needed to use a persistent triangle for remeshing.

One of the authors of this book has software available that implements the algorithm D S . There are two programs: S R [1] takes polygonal surfaces as input and remeshes them, whereas D I [2] takes voxel data as input, interpolates an isosurface over the grid, and constructs a surface mesh.

14.5 Tetrahedral meshing of volumes bounded by smooth surfaces

The beauty of using a restricted Delaunay triangulation to generate a surface mesh is that it comes with a byproduct: a Delaunay tetrahedralization of the region it bounds. The tetrahedra may have poor quality, but we can refine them, as the meshes in Figure 14.9 illustrate; observe that their boundary triangulations are the same polygonal meshes shown in Figure 14.8. This section discusses an algorithm guaranteed to produce meshes in which no tetrahedron has a radius-edge ratio greater than 2.

Tetrahedron refinement introduces vertices in the interior of the domain that can damage the surface triangulation. We solve this problem by adopting an encroachment rule similar to those used for polyhedral domains in Chapters 6 and 8. Unfortunately, curved surfaces bring a new hazard: internal vertices can be inserted too close to the boundary before the boundary is fully resolved. Imagine a concave dent in a surface. As surface refinement places new vertices on the dent, an internal vertex may become exposed because it is too close to the dent, perhaps even outside the domain. A consequence is that restricted Delaunay triangles can appear whose vertices do not all lie on the surface. We fix these triangles by deleting their internal vertices.

Let $\mathcal{O}$ be a volume enclosed by a smooth surface Σ. The definitions of restricted Voronoi diagram and restricted Delaunay triangulation work with $\mathcal{O}$ as they do for Σ, though $\mathcal{O}$ is not a surface. Thus $\mathrm{Vor}|_{\mathcal{O}} S$ is the Voronoi diagram restricted to the domain $\mathcal{O}$, and $\mathrm{Del}|_{\mathcal{O}} S$ is a subcomplex of $\mathrm{Del}\, S$ containing the simplices dual to faces of $\mathrm{Vor}|_{\mathcal{O}} S$. These include all the

[1] http://www.cse.ohio-state.edu/~tamaldey/surfremesh
[2] http://www.cse.ohio-state.edu/~tamaldey/deliso.html

simplices in $\mathrm{Del}|_\Sigma S$ and more. A tetrahedron is in $\mathrm{Del}|_\mathcal{O} S$ if its circumcenter is in $\mathcal{O}$. The algorithm does not need to keep track of dangling simplices, so we define the *restricted Delaunay* 3-*subcomplex*, denoted

$$\mathrm{Del}|^3_\mathcal{O} S = \{\tau : \tau \text{ is a face of a tetrahedron in } \mathrm{Del}|_\mathcal{O} S\}.$$

When refinement is done, the tetrahedral meshing algorithm returns the mesh $\mathrm{Del}|^3_\mathcal{O} S$.

The *boundary complex* of $\mathrm{Del}|^3_\mathcal{O} S$, written $\partial\mathrm{Del}|^3_\mathcal{O} S$, is the subcomplex containing all the triangles that are faces of exactly one tetrahedron in $\mathrm{Del}|^3_\mathcal{O} S$, and the faces of those triangles. It is easy to see that $\partial\mathrm{Del}|^3_\mathcal{O} S \subseteq \mathrm{Del}|^2_\Sigma S$, because every triangle in $\partial\mathrm{Del}|^3_\mathcal{O} S$ dualizes to a Voronoi edge that has one vertex in $\mathcal{O}$ and one outside, and therefore intersects Σ. However, $\mathrm{Del}|^2_\Sigma S$ can contain additional triangles whose dual Voronoi edges intersect Σ tangentially or at an even number of points, so we refine these triangles by inserting vertices at those intersection points. This refinement is essentially a special case of using vertices returned by the subroutine V E to refine. Ideally, we would like to have $\partial\mathrm{Del}|^3_\mathcal{O} S = \mathrm{Del}|^2_\Sigma S$, and for both to be a triangulation of the surface Σ.

The algorithm begins by computing a persistent triangle. Just like D S , it refines triangles in $\mathrm{Del}|^2_\Sigma S$ by inserting the centers of their largest surface Delaunay balls. The presence of the persistent triangle ensures that surface refinement never gets stuck. The persistent triangle is deleted as described in Section 14.1.2 once a surface mesh is built. It is important to delete the persistent triangle before refining tetrahedra; otherwise, unnecessarily small tetrahedra will be generated around its vertices.

Tetrahedra whose radius-edge ratios exceed $\bar{\rho}$ are split by new vertices inserted at their circumcenters, but a circumcenter that lies inside a surface Delaunay ball is rejected, and an encroached surface triangle is split instead. The quality threshold $\bar{\rho}$ must be at least 2 to guarantee that the algorithm terminates. Pseudocode for the main algorithm follows.

D T S $(\Sigma, \lambda, \bar{\rho})$

1. Compute a persistent triangle with vertices on Σ. Let S be the set of vertices of the persistent triangle. Compute $\mathrm{Del}\, S$ and $\mathrm{Del}|^2_\Sigma S$.
2. While $\mathrm{Del}|^2_\Sigma S$ has a vertex $p \notin \Sigma$, delete p from S, update $\mathrm{Del}\, S$, $\mathrm{Del}|^2_\Sigma S$, and $\mathrm{Del}|^3_\mathcal{O} S$, and repeat step 2.
3. If some triangle in $\mathrm{Del}|^2_\Sigma S$ has a surface Delaunay ball $B(c, r)$ with $r > \lambda(c)$, insert c into S, update $\mathrm{Del}\, S$, $\mathrm{Del}|^2_\Sigma S$, and $\mathrm{Del}|^3_\mathcal{O} S$, and go to step 2.
4. If some triangle σ in $\mathrm{Del}|^2_\Sigma S$ is a face of zero or two tetrahedra in $\mathrm{Del}|^3_\mathcal{O} S$, insert $c_{\max}(\sigma)$ into S, update $\mathrm{Del}\, S$, $\mathrm{Del}|^2_\Sigma S$, and $\mathrm{Del}|^3_\mathcal{O} S$, and go to step 2.
5. For every $p \in S$, compute $c \leftarrow$ T D (p); if some c is non-null, stop looping, insert c into S, update $\mathrm{Del}\, S$, $\mathrm{Del}|^2_\Sigma S$, and $\mathrm{Del}|^3_\mathcal{O} S$, and go to step 2.
6. If S contains more than one vertex of the persistent triangle, delete one, update $\mathrm{Del}\, S$, $\mathrm{Del}|^2_\Sigma S$, and $\mathrm{Del}|^3_\mathcal{O} S$, and go to step 2.
7. If there is a tetrahedron $\tau \in \mathrm{Del}|^3_\mathcal{O} S$ for which $\rho(\tau) > \bar{\rho}$, then
   (a) If the circumcenter z of τ lies inside a surface Delaunay ball $B(c, r)$, insert c into S, update $\mathrm{Del}\, S$, $\mathrm{Del}|^2_\Sigma S$, and $\mathrm{Del}|^3_\mathcal{O} S$, and go to step 2.

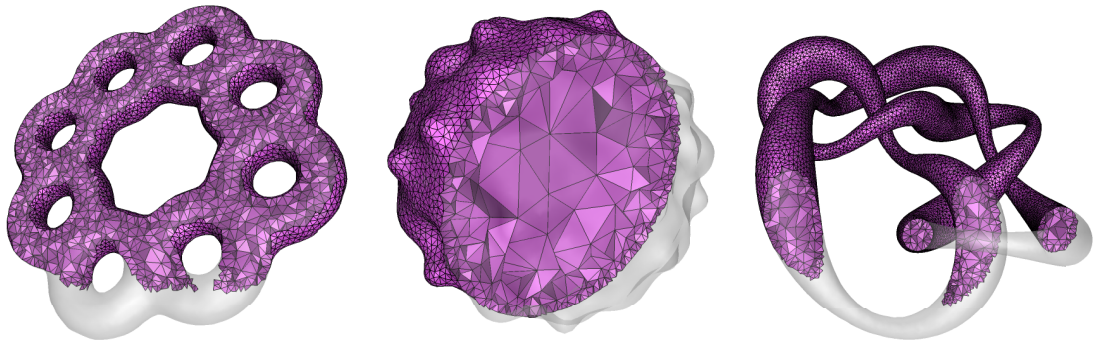


Figure 14.10: More tetrahedral meshes generated by DTS.

(b) Insert z into S and update $\mathrm{Del}\,S$, $\mathrm{Del}|^2_\Sigma S$, and $\mathrm{Del}|^3_\mathcal{O} S$.

(c) If the insertion of z creates a new triangle in $\mathrm{Del}|^2_\Sigma S$, let $B(c,r)$ be the largest new surface Delaunay ball, delete z from S, insert c into S, and go to step 2.

(d) Go to step 7.

8. Return $\mathrm{Del}|^3_\mathcal{O} S$.

Occasionally, the refinement of a boundary triangle can expose an internal vertex, making it become a vertex of $\mathrm{Del}|^2_\Sigma S$; when this happens, step 2 purges all such internal vertices before any further refinement can occur. Step 3 refines oversized surface triangles as DS does. Step 4 refines any surface triangle in $\mathrm{Del}|^2_\Sigma S$ that is missing from $\partial\mathrm{Del}|^3_\mathcal{O} S$. Step 5 ensures that $\mathrm{Del}|^2_\Sigma S$ is a 2-manifold. Step 6 deletes vertices of the persistent triangle, which could be unnecessarily close to each other. Step 7 refines skinny tetrahedra, but if a tetrahedron circumcenter threatens to delete or create a triangle in $\mathrm{Del}|^2_\Sigma S$, the circumcenter is rejected and a restricted Delaunay triangle is split instead. Step 7(a) splits restricted Delaunay triangles that are encroached upon by rejected circumcenters. Step 7(c) identifies circumcenters whose insertion creates new restricted Delaunay triangles without deleting any old ones; this occurs only when the surface mesh is coarse and the shape of Σ is still being discovered.

Figure 14.10 shows more volume meshes generated by DTS.

DTS raises the question of how to identify which tetrahedra in $\mathrm{Del}\,S$ are in $\mathrm{Del}|^3_\mathcal{O} S$, which is necessary for step 7. These tetrahedra are the ones whose dual Voronoi vertices are in $\mathcal{O}$. Recall from Section 14.1.3 that there are at least two different ways to represent a surface: as a collection of patches or polygons, or as an isosurface.

Suppose the input is a set of patches whose underlying space is a 2-manifold Σ without boundary, enclosing the volume $\mathcal{O}$ we wish to mesh. The edges of $\mathrm{Vor}\,S$ form a graph whose vertices we wish to label as being either inside the volume—these vertices dualize to tetrahedra in $\mathrm{Del}|^3_\mathcal{O} S$—or outside the volume. Initially, S contains only the three vertices of the persistent triangle, and $\mathrm{Vor}\,S$ contains only one vertex—the vertex at infinity—which is labeled "outside." When a new vertex is inserted into S, we compute labels for the new Voronoi vertices. If one vertex of a Voronoi edge is labeled and the other is not, we can normally compute the label for the second vertex by counting the number of points where the Voronoi edge intersects the patches. However,

tangential intersections are tricky because the edge may or may not cross from outside to inside the volume. One way to simplify this problem is to use the subroutine V E to refine these intersections away, replacing step 4.

Isosurfaces, by contrast, make the identification of $\mathrm{Del}|^3_{\mathcal{O}} S$ easy—even easier than identifying $\mathrm{Del}|^2_{\Sigma} S$. Recall that an isosurface of a function $h : \mathbb{R}^3 \to \mathbb{R}$ is a level set $\Sigma = h^{-1}(\eta)$ for a selected isovalue η. Suppose that h is greater than η inside the volume to be meshed, hence $\mathcal{O} = \{p : h(p) \geq \eta\}$. Then determining whether the dual of a Voronoi vertex v is a restricted Delaunay tetrahedron is a simple matter of evaluating $h(v)$.

For an isosurface, computing $\mathrm{Del}|^2_{\Sigma} S$ is more difficult than computing $\mathrm{Del}|^3_{\mathcal{O}} S$, as it necessitates extra computation to identify Voronoi edges that cross the isosurface an even number of times. It is natural to ask whether these computations can be eliminated by replacing $\mathrm{Del}|^2_{\Sigma} S$ with $\partial\mathrm{Del}|^3_{\mathcal{O}} S$ in D T S (and deleting step 4). This question remains open (see Exercise 9), although it is clear that the persistent triangle must be replaced with a stronger strategy; $\partial\mathrm{Del}|^3_{\mathcal{O}} S$ is empty when S contains only the vertices of the persistent triangle. It is always possible to fall back on edge-surface intersection computations when $\partial\mathrm{Del}|^3_{\mathcal{O}} S$ is empty.

14.5.1 Proof of termination and guarantees

As usual, our goal is to prove that there is a positive lower bound on the distances among the vertices, hence D T S terminates, hence it returns a mesh in which every tetrahedron has a radius-edge ratio at most $\bar{\rho}$.

The size field λ and local feature size function f are defined only on Σ. We define a function that extends λ's domain over the entirety of $\mathcal{O}$, and forces it to be $(\bar{\rho}-2)/(3\bar{\rho})$-Lipschitz everywhere in $\mathcal{O}$. Let $\mu : \mathcal{O} \to \mathbb{R}$ be the regularized size field

$$\mu(x) = \inf_{y \in \Sigma} \left(\min\{\lambda(y), 0.08\, f(y)\} + \frac{\bar{\rho} - 2}{3\bar{\rho}} d(x, y) \right),$$

and observe that $\mu(x) \leq \lambda(x)$ and $\mu(x) \leq 0.08\, f(x)$ for every point $x \in \Sigma$. Note that we use μ solely to analyze the algorithm, not as an input. As in Section 14.4, the Lipschitz property captures how refining for quality affects the edge lengths. The more $\bar{\rho}$ exceeds 2, the more strongly the mesh can be graded.

Recall from Section 8.3 that the *insertion radius* r_x of a vertex x is the distance from x to the nearest distinct vertex in the sample S when x is inserted into S. Immediately after its insertion, r_x is the length of the shortest edge adjoining x in $\mathrm{Del}\, S$.

Proposition 14.12. *Let S be the set of vertices in a mesh returned by* D T S $(\Sigma, \lambda, \bar{\rho})$. *If $\bar{\rho} \geq 2$, then every vertex $p \in S$ has insertion radius $r_p > \mu(p)$, and every internal vertex $p \in S \setminus \Sigma$ has insertion radius $r_p > \mu(p) \cdot 3\bar{\rho}/(\bar{\rho} + 1)$.*

P . When step 3 of D T S inserts a vertex c, it is at the center of a surface Delaunay ball whose interior contains no vertex and whose radius exceeds $\lambda(c)$. Therefore, $r_c > \lambda(c) \geq \mu(c)$.

When step 4 of D T S inserts a vertex c, it is at the center of a surface Delaunay ball of a triangle $\sigma \in \mathrm{Del}|^2_{\Sigma} S$ that is a face of either zero or two tetrahedra in $\mathrm{Del}|^3_{\mathcal{O}} S$. Therefore, the vertices of its dual Voronoi edge σ^* are either both in $\mathcal{O}$ or both outside $\mathcal{O}$, so σ^* intersects Σ either tangentially or at more than one point. The new vertex c is the same vertex that would

be returned by V E (σ^*). Let p be a vertex of σ; thus σ^* is an edge of the Voronoi cell V_p. By the same reasoning given in the proof of Proposition 14.6, $d(c,p) > 0.15f(p)$. Therefore, $r_c > 0.15f(p) > 0.13f(c) \geq \mu(c)$ by the Feature Translation Lemma.

When step 5 of D T S inserts a vertex c, some vertex p adjoins a nonempty set of triangles in $\mathrm{Del}^2_\Sigma S$ that do not form a topological disk. So Σ_p is nonempty. Σ_p is not 0.09-small; otherwise, the Small Intersection Theorem (Theorem 13.6) would imply that the triangles adjoining p form a topological disk. It follows that some triangle adjoining p has a surface Delaunay ball with radius $d(c,p) > 0.09f(p) > 0.08f(c)$ by the Feature Translation Lemma. Therefore, $r_c > 0.08f(c) \geq \mu(c)$.

We use induction to show that a vertex inserted by step 7 also satisfies the claims. Assume that $r_p > \mu(p)$ for every previously inserted vertex p. This assumption holds before the first tetrahedron is refined because steps 3, 4, and 5 ensure it for every vertex except the vertices of the persistent triangle, two of which are deleted by step 6. Step 7 inserts a vertex either at the circumcenter z of a tetrahedron τ whose radius-edge ratio exceeds $\bar{\rho}$, or at the center c of a surface Delaunay ball that contains z. The circumradius of τ is $R_\tau > \bar{\rho}\ell$, where ℓ is the length of the shortest edge of τ. Let p be the most recently inserted vertex of that shortest edge. Then $r_p \leq \ell$ and

$$R_\tau > \bar{\rho} r_p \geq \bar{\rho} \cdot \mu(p).$$

As μ is $(\bar{\rho}-2)/(3\bar{\rho})$-Lipschitz,

$$\begin{aligned} R_\tau &> \bar{\rho}\left(\mu(z) - \frac{\bar{\rho}-2}{3\bar{\rho}} d(p,z)\right) \\ &= \bar{\rho}\left(\mu(z) - \frac{\bar{\rho}-2}{3\bar{\rho}} R_\tau\right) \\ \Rightarrow R_\tau &> \frac{3\bar{\rho}}{\bar{\rho}+1}\mu(z). \end{aligned} \tag{14.2}$$

If z does not encroach upon a surface Delaunay ball, then step 7(b) inserts z and $r_z = R_\tau > \mu(z) \cdot 3\bar{\rho}/(\bar{\rho}+1)$, as claimed.

Suppose the circumcenter z of τ encroaches upon a surface Delaunay ball B, and step 7(a) inserts a vertex c at its center; or the insertion of z creates a new surface Delaunay ball B, and step 7(c) inserts a vertex c at its center. In either case, there is no vertex in B's interior, so r_c is the radius of B. Because $z \in B$, $d(c,z) \leq r_c$, so

$$\begin{aligned} \mu(z) &\geq \mu(c) - \frac{\bar{\rho}-2}{3\bar{\rho}} d(c,z) \\ &\geq \mu(c) - \frac{\bar{\rho}-2}{3\bar{\rho}} r_c \end{aligned} \tag{14.3}$$

The ball B circumscribes some triangle in $\mathrm{Del}^2_S \Sigma$. Let p be a vertex of this triangle other than z. Observe that p lies on the boundary of B, but not in the open circumball of the Delaunay

tetrahedron τ. As $z \in B$,

$$\begin{aligned} 2r_c &\geq d(p,z) \\ &\geq R_\tau \\ &\overset{(14.2)}{>} \frac{3\bar{\rho}}{\bar{\rho}+1}\,\mu(z) \\ 2(\bar{\rho}+1)r_c &\overset{(14.3)}{\geq} 3\bar{\rho}\cdot\mu(c) - (\bar{\rho}-2)r_c. \end{aligned}$$

Rearranging terms gives $r_c > \mu(c)$. □

Proposition 14.13. *If $\bar{\rho} \geq 2$ and $\inf_{x\in\Sigma} \lambda(x) > 0$,* D T S *$(\Sigma, \lambda, \bar{\rho})$ terminates and returns a mesh in which every vertex p is at a distance greater than $\mu(p)\cdot 3\bar{\rho}/(4\bar{\rho}-2)$ from the nearest distinct vertex.*

P . Observe that $\mu_{\min} = \inf_{x\in\mathcal{O}} \mu(x)$ is strictly positive because $\inf_{x\in\Sigma} \lambda(x)$ and $\inf_{x\in\Sigma} f(x)$ are. By Proposition 14.12, every vertex p is at a distance greater than $\mu(p)$ from every vertex inserted before it. For any vertex q inserted after p, $d(p,q) > \mu(q) \geq \mu(p) - d(p,q)\cdot(\bar{\rho}-2)/(3\bar{\rho})$. Rearranging terms gives $d(p,q) > \mu(p)\cdot 3\bar{\rho}/(4\bar{\rho}-2)$.

By the Packing Lemma (Lemma 6.1), there is an upper bound on the number of vertices in the mesh. The Packing Lemma alone does not guarantee termination, because vertices are sometimes deleted. Step 2 can delete internal vertices, but only vertices of $\mathrm{Del}|^2_\Sigma S$. Step 7 never inserts an internal vertex that changes $\mathrm{Del}|^2_\Sigma S$; an internal vertex can become part of $\mathrm{Del}|^2_\Sigma S$ only during the insertion of a new surface vertex. Therefore, after step 2 finishes looping, it cannot run again until at least one new surface vertex is inserted. Surface vertices, except two vertices of the persistent triangle, are never deleted. Eventually there will be no room for new surface vertices, then no room for new internal vertices, whereupon D T S terminates. □

D T S returns the tetrahedral subcomplex $\mathrm{Del}|^3_\mathcal{O} S$ and enforces the requirement that $\partial\mathrm{Del}|^3_\mathcal{O} S = \mathrm{Del}|^2_\Sigma S$, regardless of the size field λ. If λ is sufficiently small, the two complexes coincide with the subcomplex $\mathrm{Del}|^2_\Sigma (S\cap\Sigma)$ obtained by discarding the vertices not on the surface.

Proposition 14.14. *After* D T S *terminates, the following statements hold.*

(i) $\partial\mathrm{Del}|^3_\mathcal{O} S = \mathrm{Del}|^2_\Sigma S$.

(ii) *If $\lambda(x) \leq 0.08 f(x)$ for every $x \in \Sigma$, then $\partial\mathrm{Del}|^3_\mathcal{O} S = \mathrm{Del}|^2_\Sigma S = \mathrm{Del}|^2_\Sigma (S\cap\Sigma) = \mathrm{Del}|_\Sigma (S\cap\Sigma)$, and all four have an underlying space homeomorphic to Σ.*

P . (i) D T S terminates only if every triangle in $\mathrm{Del}|^2_\Sigma S$ is in $\partial\mathrm{Del}|^3_\mathcal{O} S$ too. Every triangle σ in $\partial\mathrm{Del}|^3_\mathcal{O} S$ is a face of exactly one tetrahedron in $\mathrm{Del}|^3_\mathcal{O} S$, so the Voronoi edge dual to σ has one vertex in $\mathcal{O}$ and one outside. Hence the Voronoi edge intersects Σ, and σ is in $\mathrm{Del}|^2_\Sigma S$.

(ii) Any time execution reaches step 7—that is, just before tetrahedron refinement begins or resumes—the restricted Delaunay 2-subcomplex is a triangulation of Σ by Theorem 14.2. As $\lambda(x) \leq 0.08 f(x)$, Σ_p is 0.09-small for every vertex p in the mesh. By the Dense Sample Theorem (Theorem 13.15), every edge in $\mathrm{Vor}\,(S\cap\Sigma)$ that intersects Σ does so transversally at a single point,

and by the Homeomorphism Theorem (Theorem 13.16), $|\mathrm{Del}|_\Sigma(S \cap \Sigma)|$ is homeomorphic to Σ, which implies that $\mathrm{Del}|^2_\Sigma(S \cap \Sigma) = \mathrm{Del}|_\Sigma(S \cap \Sigma)$.

Consider the Voronoi diagram with the interior vertices as well. Steps 2 and 7 of the algorithm ensure that when D T S terminates, every triangle in $\mathrm{Del}|^2_\Sigma S$ has all three of its vertices on Σ. Therefore, every edge in $\mathrm{Vor}\, S$ that intersects Σ is generated by vertices on Σ, and is thus included in some edge of $\mathrm{Vor}(S \cap \Sigma)$, and thus intersects Σ transversally at a single point. It follows that $\mathrm{Del}|^2_\Sigma S \subseteq \mathrm{Del}|^2_\Sigma(S \cap \Sigma)$.

If the inclusion $\mathrm{Del}|^2_\Sigma S \subset \mathrm{Del}|^2_\Sigma(S \cap \Sigma)$ is strict, then some edge in $\mathrm{Del}|^2_\Sigma S$ is an edge of a single triangle, because $|\mathrm{Del}|^2_\Sigma(S \cap \Sigma)|$ is a connected 2-manifold without boundary. But $\partial\mathrm{Del}|^3_\mathcal{O} S$ cannot contain an edge of exactly one triangle, so $\mathrm{Del}|^2_\Sigma S = \mathrm{Del}|^2_\Sigma(S \cap \Sigma)$. □

We summarize the properties of meshes generated by D T S .

Theorem 14.15. *Let $\mathcal{O}$ be a volume bounded by a C^2-smooth, compact, connected surface $\Sigma \subset \mathbb{R}^3$ without boundary. Suppose $\inf_{x\in\Sigma} \lambda(x) > 0$ and $\bar{\rho} \geq 2$. For properties (iv)–(vi) below, suppose also that $\lambda(x) \leq \varepsilon f(x)$ for some $\varepsilon \leq 0.08$ and every $x \in \Sigma$. Then* D T S *$(\Sigma, \lambda, \bar{\rho})$ terminates and returns a mesh $\mathcal{T} = \mathrm{Del}|^3_\mathcal{O} S$ with the following properties.*

(i) *$|\mathcal{T}|$ is a 3-manifold whose boundary is a 2-manifold without boundary.*

(ii) *No tetrahedron in $\mathcal{T}$ has a radius-edge ratio exceeding $\bar{\rho}$.*

(iii) *Every edge adjoining each vertex p is longer than $\mu(p) \cdot 3\bar{\rho}/(4\bar{\rho} - 2)$.*

(iv) *The underlying space of $\mathcal{T}$ is homeomorphic to $\mathcal{O}$, and the two are related by an isotopy.*

(v) *The boundary complex of $\mathcal{T}$ is $\partial\mathcal{T} = \mathrm{Del}|^2_\Sigma S = \mathrm{Del}|_\Sigma(S \cap \Sigma)$.*

(vi) *The Hausdorff distance between $|\mathcal{T}|$ and $\mathcal{O}$ is less than $15\frac{\varepsilon^2}{(1-\varepsilon)^2} f_{\max}$, where $f_{\max} = \sup_{p\in\Sigma} f(p)$.*

P . (i) By Proposition 14.13, D T S terminates. At that time, $|\mathrm{Del}|^2_\Sigma S|$ is a 2-manifold without boundary because T D does not permit termination otherwise. By Proposition 14.14(i), the boundary complex is $\partial\mathcal{T} = \partial\mathrm{Del}|^3_\mathcal{O} S = \mathrm{Del}|^2_\Sigma S$. The space bounded by a 2-manifold embedded in $\mathbb{R}^3$ is a 3-manifold.

(ii, iii) D T S terminates only when no tetrahedron has a radius-edge ratio exceeding $\bar{\rho}$. The bound on edge lengths is taken from Proposition 14.13.

(iv, v) By Proposition 14.14, $\partial\mathcal{T} = \partial\mathrm{Del}|^3_\mathcal{O} S = \mathrm{Del}|^2_\Sigma S = \mathrm{Del}|_\Sigma(S \cap \Sigma)$ has an underlying space homeomorphic to Σ. As discussed in the proof of that proposition, Σ_p is 0.09-small for every $p \in S \cap \Sigma$, so the Surface Discretization Theorem (Theorem 13.22) implies that $|\mathrm{Del}|_\Sigma(S \cap \Sigma)|$ and Σ are related by an ambient isotopy. Specifically, Proposition 13.20 constructs an ambient isotopy $\xi : \mathbb{R}^3 \times [0, 1] \to \mathbb{R}^3$ such that $\xi(|\partial\mathcal{T}|, 0) = |\partial\mathcal{T}|$ and $\xi(|\partial\mathcal{T}|, 1) = \Sigma$. Because $|\partial\mathcal{T}|$ and Σ are both connected 2-manifolds without boundary that partition $\mathbb{R}^3$ into two pieces, the map $\xi(\cdot, 1)$ is a homeomorphism between the regions they enclose, $|\mathcal{T}|$ and $\mathcal{O}$. Moreover, ξ is an ambient isotopy that takes the identity map $\xi(\cdot, 0)|_{|\mathcal{T}|} : |\mathcal{T}| \to |\mathcal{T}|$ to the homeomorphism $\xi(\cdot, 1)|_{|\mathcal{T}|} : |\mathcal{T}| \to \mathcal{O}$.

(vi) The Hausdorff distance between $|\mathcal{T}|$ and $\mathcal{O}$ is realized by points on their boundaries, and therefore equals the Hausdorff distance between $|\mathrm{Del}|_\Sigma(S \cap \Sigma)|$ and Σ. With $\lambda(x) \leq \varepsilon f(x)$ for some $\varepsilon \leq 0.08$, Σ_p is $\frac{\varepsilon}{1-\varepsilon}$-small for every $p \in \Sigma$. By the Surface Discretization Theorem, the Hausdorff

distance between $|\mathrm{Del}|_\Sigma (S \cap \Sigma)|$ and Σ is less than $15\frac{\varepsilon^2}{(1-\varepsilon)^2} f_{\max}$. □

Recall from Section 14.5 that if Σ is an isosurface, DelaunayTriangulationSurfaceMesher becomes simpler and faster if we replace $\mathrm{Del}|^2_\Sigma S$ with $\partial\mathrm{Del}|^3_{\mathcal{O}} S$, and do not compute $\mathrm{Del}|^2_\Sigma S$ at all. Unfortunately, we do not know whether the guarantees (iv)–(vi) of Theorem 14.15 hold for $\partial\mathrm{Del}|^3_{\mathcal{O}} S$ when λ is sufficiently small, although it seems likely to be true. We leave it as an open question in Exercise 9.

14.6 Notes and exercises

Early surface meshing algorithms developed for computer graphics and medical imaging were not concerned with triangle quality. By far the most famous and heavily used of these is the 1987 *marching cubes* algorithm of Lorensen and Cline [141], from the most-cited paper in the history of the conference SIGGRAPH. The marching cubes algorithm triangulates an isosurface of a function h by computing the value of h at each vertex of a cubical grid, then the intersections of the isosurface with the edges of the grid, then a few triangles in each cube that span the intersection points. It can also create isosurfaces from voxel data such as medical images, in which case the intersections of an unknown surface with the grid edges can be estimated by linear interpolation. The algorithm is very fast, and performs interpolation only over edges, not over whole cubes. Unfortunately, it can create arbitrarily skinny triangles. Bloomenthal [24] proposes a similar method that uses a tetrahedral background grid.

After the development of Delaunay refinement meshing in the late 1980s, Chew [61] in 1993 presented an algorithm for triangular surface meshing that flips the edges of an existing surface triangulation so that if its vertices are sufficiently dense, the final triangulation is, in some sense, Delaunay. Then the algorithm refines the mesh by inserting vertices at the centers of surface Delaunay balls of poor-quality triangles, until no triangle has an angle less than 30°. In retrospect, the meshes this algorithm generates are usually subsets of a restricted Delaunay triangulation.

The introduction of restricted Delaunay triangulations by Edelsbrunner and Shah [92], their Topological Ball Theorem, and the sampling theory of Amenta and Bern [3] opened a path to developing algorithms with topological guarantees. By connecting Chew's refinement strategy with the ε-sampling theory, Cheng, Dey, Edelsbrunner, and Sullivan [47] give a Delaunay refinement algorithm for generating a restricted Delaunay triangulation that is homeomorphic to a specialized smooth surface for molecular modeling, called a *skin surface* [86].

Boissonnat and Oudot [29, 30] present a Delaunay refinement algorithm for homeomorphic meshing of a more general class of smooth surfaces. Our DelaunaySurfaceMesher1 pseudocode is essentially this algorithm without support for surface boundaries, which we address in the next chapter. Dey and Levine [76] show that once the sample is sufficiently dense, one may discard the tetrahedralization and continue to refine the surface triangulation; see Exercise 3.

Cheng, Dey, Ramos, and Ray [53] propose a topology-driven surface meshing algorithm that uses critical point computations and the TangentDelaunay subroutine. Violations of the topological ball property are detected and repaired. The algorithm does not need to estimate local feature sizes. This algorithm is embodied in our DelaunaySurfaceMesher2 pseudocode.

Our DelaunayTriangulationSurfaceMesher pseudocode is a variation of the tetrahedral mesh generation algorithm of Oudot, Rineau, and Yvinec [162] for volumes bounded by smooth surfaces. We have modified it by using vertex deletions, thereby improving the radius-edge ratio bounds, permitting the algo-

rithm to work reasonably well with sparse samples, and removing the requirement for a Lipschitz size field. The value of vertex deletion for improving both the theoretical guarantees and the practical performance of Delaunay refinement algorithms was introduced by Chew's pioneering paper [61].

A related tetrahedral meshing algorithm called *variational tetrahedral meshing* combines Delaunay triangulations and vertex smoothing in an interesting way. Recall that the Delaunay triangulation of a fixed vertex set minimizes the volume bounded between the triangulation lifted to the parabolic lifting map and the paraboloid itself, compared with all other triangulations of the same vertex set. What if the domain's interior vertices are not fixed? Chen and Xu [46] suggest globally smoothing the interior vertices by minimizing the same volume. Alliez, Cohen-Steiner, Yvinec, and Desbrun [2] implemented an iterative tetrahedral mesh improvement method that alternates between this global smoothing step and recomputing the Delaunay triangulation—in other words, it alternates between numerical and combinatorial optimization of the triangulation with respect to the same objective function. During smoothing steps, vertices on the boundary are constrained to maintain domain conformity.

A shortcoming of Delaunay refinement is that it does not scale well to meshes that are too large to fit in main memory, because its random patterns of access to a mesh can cause virtual memory to thrash. Dey, Levine, and Slatton [78, 80] study how to make algorithms in this chapter more local in their data access patterns so that they can generate huge meshes without heavy thrashing.

Although most guaranteed-quality algorithms for meshing surfaces and the volumes they enclose are founded on Delaunay triangulations, an exception is the *isosurface stuffing* algorithm of Labelle and Shewchuk [128], which uses an octree to fill a smooth isosurface with tetrahedra whose dihedral angles are guaranteed to be bounded between 10.7° and 164.8°. Unfortunately, no known Delaunay algorithm offers similar guarantees in theory, although they often achieve better angles in practice.

There are many non-Delaunay surface meshing methods that focus on obtaining correct topology, but not element quality, especially for isosurfaces and implicit surfaces. A few examples are the small normal variation method of Plantinga and Vegter [171], the sweep approach of Mourrain and Técourt [153], the adaptive approach of Snyder [207], and the critical point method of Boissonnat, Cohen-Steiner and Vegter [28].

Isosurfaces of smooth functions can be nonmanifold and nonsmooth for isovalues that are critical values. Such isosurfaces cannot be meshed by the methods described in this chapter, but recent research [138, 153] has begun to address this problem.

Edge-surface intersection computations are expensive. Boissonnat and Oudot [30] observe an idea for reducing the number of these computations when the surface Σ is an isosurface of a function h. Boissonnat and Oudot show that, instead of maintaining $\mathrm{Del}|^2_\Sigma S$, D S 1 can maintain a complex containing only the Delaunay triangles dual to the bipolar Voronoi edges, and compute only a single intersection point (and a single surface Delaunay ball) for each such edge, thereby reducing the computation time considerably.

Implementations of algorithms involving curved surfaces are plagued by floating-point roundoff error, which often causes algorithms to fail. Researchers are making geometric algorithms more numerically robust with sophisticated methods from computational real algebraic geometry. The literature on geometric robustness is too large to survey here. The software produced by the

CGAL project[3] is making continuing progress in robust computing with curved geometries.

Exercises

1. [30] Let $\mathcal{B}$ be the set of surface Delaunay balls for a sample on a smooth surface Σ without boundary. Prove that Σ is included in the union of the balls in $\mathcal{B}$ if each ball with circumcenter c has a circumradius of at most $0.09 f(c)$.

2. Suppose we initialize D S 1 with a set of n random points on Σ instead of the vertices of a persistent triangle. Show that the algorithm might be unable to insert additional vertices by showing that for every n, there is a surface and a sample of size n such that no Voronoi edge intersects Σ.

3. [76] Suppose we have computed $\mathrm{Del}|_\Sigma S$ for a 0.09-sample S of a smooth surface Σ without boundary. Design an algorithm that can insert the center of a surface Delaunay ball into S and update $\mathrm{Del}|_\Sigma S$ without accessing the three-dimensional Delaunay triangulation $\mathrm{Del}\, S$. Then design a Delaunay refinement procedure for making a 0.09-sample even denser without computing $\mathrm{Del}\, S$.

4. Suppose that Σ is given as an isosurface $h(x, y, z) = 0$. Explain in detail the numerical computations needed to compute the critical set $X \cap g$ in step 4 of the subroutine V F .

5. [53] Explain in detail the numerical computations needed for S P and S - C .

6. Some of the meshing algorithms in this chapter use vertex deletions. Suppose that the Voronoi diagram $\mathrm{Vor}\, S$ satisfies the TBP for a sample S of a smooth surface. Prove that for every site $s \in S$ and for every site $p \in S \setminus \{s\}$, each restricted Voronoi cell $V_p \cap \Sigma$ in $\mathrm{Vor}\,(S \setminus \{s\})$ remains a topological disk. (Note that this does not imply that the TBP still holds.)

7. Theorem 14.15 proves that the Hausdorff distance between $|\mathrm{Del}^3_{\mathcal{O}} S|$ and $\mathcal{O}$ is small. Prove that under the same conditions, there is an isotopy relating $|\mathrm{Del}^3_{\mathcal{O}} S|$ to $\mathcal{O}$ that moves each point $x \in |\mathrm{Del}^3_{\mathcal{O}} S|$ by a distance less than $15\varepsilon^2 f(x)$.

8. Suppose we modify D T S as follows. Explain what happens in each case.

   (a) Exclude step 6. Theorem 14.15 needs to be changed in this case.

   (b) Exclude step 7(c). The algorithm might not terminate; why?

9. Recall that if Σ is an isosurface, it is desirable to replace $\mathrm{Del}^2_\Sigma S$ with $\partial\mathrm{Del}^3_{\mathcal{O}} S$ in D T - S . Also recall that $\partial\mathrm{Del}^3_{\mathcal{O}} S \subseteq \mathrm{Del}^2_\Sigma S$. Suppose the modified algorithm is started with a point sample $S \subset \Sigma$ such that while it runs, $\partial\mathrm{Del}^3_{\mathcal{O}} S$ never becomes empty. Suppose that $\lambda(x) \leq 0.08 f(x)$ for each $x \in \Sigma$. It is an open problem whether the properties (iv)–(vi) of Theorem 14.15 hold for the modified algorithm. Prove or disprove them.

[3] http://www.cgal.org