Efficient Mesh Generation for Piecewise Linear Complexes

Todd Phillips

CMU-CS-09-156

August 2009

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Gary L. Miller, Chair
Anupam Gupta
Daniel D.K. Sleator
Noel J. Walkington
Tamal K. Dey, The Ohio State University

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This work was supported in part by the National Science Foundation under grant CCF-0635257

Keywords: Computational Geometry, Scientific Computing, Mesh Generation

Abstract

The mesh generation problem is to output a set of tetrahedra that discretize an input geometry. The input is given as a piecewise linear complex (PLC), a set of points, lines, and polygons to which the output tetrahedra must conform. Additionally, a mesh generation algorithm must make guarantees on the quality and number of output tetrahedra. Downstream applications in scientific computing and visualization necessitate these guarantees on the mesh.

Recent advances have led to provably correct algorithms for a number of input classes. Particular difficulties arise when the input contains *creases*, regions where input segments or polygons meet at acute angles. When the input is *without* creases, the mesh generation problem is better understood. Algorithms for such inputs exist with near-optimal runtimes of $O(n \log \Delta + m)$, where n and m are the size of the input and output, and Δ is the ratio of largest-to-smallest distances in the input geometry. The principle result of this thesis is to extend this result to the general case of piecewise linear complexes *with* creases.

Correct algorithms to handle inputs with creases involve explicitly constructing a system of specially designed *collars* around the creases. These collars must be specifically sized according to the input geometry. I give a new procedure to compute the needed collar sizes in near-optimal $O(n \log \Delta + c)$, where c is the description complexity of the collar system. Additionally, I give a procedure for *implicitly* constructing a collar system on the fly, so that a complete meshing algorithm for a PLC can be run in one pass with total work in $O(n \log \Delta + m)$ and space usage in $O(m)$.

Central to the analysis is the "Scaffold-Sizing Theorem", a structural result governing the number of vertices created during mesh generation. The theorem is general enough to have an added benefit of retroactively improving the analysis of almost all existing meshing algorithms.

Contents

Chapter 1

Introduction

1.1 The Meshing Problem

Througout the last 60 years, problems in engineering and scientific computation continually seek to ask questions about the physical properties of a given object: *What is the airflow past this airplane wing?, How does heat diffuse in this reactor bed?* ; even questions as simple as visualization: *What does this scene look like from above?* All of these problems require calculations based on the geometry of the input being considered. Computations on arbitrarily complex geometries quickly become arbitrarily lengthy, and so we arrive at the science of mesh generation; approximating a geometry for computational needs.

One of the main difficulties in handling the problem of mesh generation is that it is by nature a very interdisciplinary problem. Solutions must reach beyond traditional computer science algorithms research to address concerns from discrete geometry, numerical analysis, topology, engineering, and more. Accordingly, the literature is bereft of a unified answer to the perennial question, *What is the exact problem being solved?*

The simplistic answer is the following: given a shape, a mesh generation algorithm seeks to decompose it into some number of pieces that are suitable for future computations on the shape. In any medium-sized group of meshing enthusiasts, one is hard-pressed to find a less vague common denominator. But, working from this coarse problem description, I quantify the meshing problem into four distinct subproblems that are addressed by meshing solutions: conforming, element quality, sizing, and efficiency.

The problem of conforming asks first, "what is an input shape?", and secondly, "what is a decomposition of such a shape?". This is tied to element quality, which must define what is a "piece" in a decomposition, and what makes a piece suitable for the future computation. The runtime cost of downstream these computations is undoubtedly tied to the number of pieces in the decomposition, and so the problem of sizing presents itself. Beyond these three subproblems, the most interest question from a computer science perspective is that of efficiency, what guarantees do we have about the time and space usage of an

algorithm? The central concern of this thesis is a new mesh generation algorithm, `Sparse Voronoi Refinement with Collars` (`SVRC`), that contributes in all four areas, by increasing the realm of inputs that can be handled by provably efficient algorithms, and by improving the guarantees on sizing for a large class of algorithms. In the following exposition, I give an introduction to the research questions of each subproblem, discuss the state of the art, and state how `SVRC` and its analysis address each area.

This thesis is primarly concerned with meshing inputs drawn from $\mathbb{R}^3$. All of the results generalize inputs to $\mathbb{R}^2$, and some of the results, particularly those on mesh sizing, can be generalized to $\mathbb{R}^d$ for constant d. I will generally assume that meshing is needed in some closed, bounded, convex domain $\Omega \subset \mathbb{R}^3$.

For $x, y \in \mathbb{R}^d$, $|xy|$ denotes the Euclidean distance from x to y. If A is a set, then $|xA| := \min\{|xa| \mid a \in A\}$. The closed d-Ball centered at x of radius r is denoted $B_d(x, r) := \{y \in \mathbb{R}^d \mid |xy| \leq r\}$. The subscript d will normally be suppressed and all balls are 3-balls unless otherwise noted. Somewhat less frequently, B° will refer to the corresponding open ball and ∂B to the spherical boundary. The linear algebraic dimension of a set S is denoted $dim(S)$. For shorthand, the notation $A \gtrsim B$ will be used to denote the existence of a constant C such that $A \geq C \cdot B$. When using $\gtrsim$, the constant will be independent of A and B, but most likely dependent on other constants in the hypotheses. Similarly $\lesssim$ will be used and $A \sim B$ is taken to mean $A \lesssim B$ and $A \gtrsim B$. Any notable (in)dependences are mentioned. This notation provides brevity as long as care is taken with regards to combining such statements.*

.

1.2 Conforming Meshes

The principle questions posed when dealing with a conforming meshing algorithm are defining an input class and defining a notion of a conformal mesh.

The simplest input for meshing is generally considered to be a finite set of input points $N \subset \Omega$. Often in this case, the domain is considered to be periodic, looping back on itself, in order to avoid boundary concerns. When the input is a point-set, the notion of a conforming mesh is the straightforward requirement that the output have a superset of vertices M so that $N \subset M$. This stripped-down view of input often allows for fairly elegant algorithms especially for high dimensions, but does not cover many practical inputs.

The next most interesting class of input is that of a **planar straight line graph** (PSLG) in two dimensions. This consists of a set of a set of vertices and edges (closed line segments) such that two edges only intersect at vertex. The PSLG generalizes to three and higher dimensions via a piecewise linear complex [MTT+96].

*In computer science research, the notations $O(\ldots)$ and $\Omega(\ldots)$ are often abused when dealing with quantities not tending toward infinity or zero. I abhor this practice, although I am not necessarily innocent.

Definition 1 (Piecewise Linear Complex)**.** *A d-***piecewise linear complex** *(d-PLC) is a set $\mathcal{P}$ of closed* **features** *such that:*

- $\forall\ F \in \mathcal{P}$, *$F$ is a closed subset of* $\mathbb{R}^{d+1}$.
- $\forall\ F \in \mathcal{P},\ dim(F) \leq d$
- $\forall\ F \in \mathcal{P}$, *the boundary of F is a union of features of* $\mathcal{P}$
- $\forall\ F, G \in \mathcal{P},\ F \cap G \in \mathcal{P}$
- *If* $dim(F \cap G) = dim(F)$, *then* $F \subset G$ *and* $dim(F) < dim(G)$

PLC will be taken to mean a 2-PLC by default.

The terms subfeature and superfeature are used to refer to set containment. Features of dimension $0 \ldots 2$ may be called input vertices, segments, or facets. Note that any point set is a PLC. With this definition in hand, define a conforming mesh.

The output $\mathcal{T}$ of a meshing algorithm is a 3-PLC. $\mathcal{T}$ is then a conforming mesh if it covers the whole domain, and each feature from the input $\mathcal{P}$ appears as a union of subfeatures from $\mathcal{T}$. More precisely:

Definition 2 (Conforming Mesh)**.** *A 3-PLC $\mathcal{T}$ is a* **conforming mesh** *for 2-PLC $\mathcal{P}$ in domain $\Omega \subset \mathbb{R}^3$ if:*

$$\Omega = \cup\ \mathcal{T}$$

$$\forall\ F \in \mathcal{P},\ \exists\ \mathcal{F} \subset \mathcal{T},\ F = \cup \mathcal{F}$$

This notion of conforming restricts attention to algorithms that generate a volume filling mesh, since it must cover the three-dimensional Ω. Other algorithms may generate a a surface-filling mesh that only covers the input 2-PLC.

Given an input PLC, an important measure of its complexity is the **spread** Δ of the geometry, sometimes called the aspect-ratio ratio of the input or (L/s) in other works.

Definition 3 (Spread Δ)**.** *Given a set of geometric objects $\mathcal{P}$, define Δ, the* **spread** *of $\mathcal{P}$, as the ratio of the diameter of $\mathcal{P}$ to the smallest distance between two disjoint objects in $\mathcal{P}$.*

Another measure of difficulty in conforming to PSLGs and PLCs regards the angle between features. Define the angle between two intersecting linear features, following [MPW02a]:

Definition 4 (Angle Between Features)**.** *Consider a 2-PLC. Suppose two features F and G intersect at a feature H, Define rays(H, F) as the set of all rays emanating from a point in H, orthogonal to H, and heading into F. Then the* **angle between features** *F and G is the minimum angle subtended by any pair from rays(H, F) and rays(H, G).*

The definition is as such to handle the interaction between a facet and a segment or two facets intersecting at a vertex. The angle between two intersecting segments will be the usual. Between two faces intersecting along a segment, it will be the dihedral angle they subtend. This leads to the definition of **α-creases**:

Definition 5 (α-crease)**.** *Given a PLC $\mathcal{P}$, $F \in \mathcal{P}$ is an α-crease if there exist $G, H \in \mathcal{P}$ such that $F \subset G \cap H$, and the angle between G and H is less than α. When α is suppressed it is taken to be $90°$.*

For PLCs without creases in two dimensions, the earliest conforming algorithms date to Chew and Ruppert [Che89a, Rup95][†]. It was later shown that their algorithms could handle creases as small as $60°$, and variants on these algorithms exist to handle arbitrarily small creases in two dimensions [She97a, Pav03].

The conforming algorithms for PLCs in three dimensions easily handle PLCs without creases [She98, PW04]. The first correct conforming algorithm for arbitrary PLCs was given by Cheng and Poon [CP06]. Other conforming algorithms have followed [PW05, RW08]. All of the algorithms for arbitrary PLCs in two and three dimensions have a similar flavor; creases are protected by a system of **collars** (sometimes "intestines"), and special procedures are explicitly utilized for meshing within and around the collars.

The `SVRC` algorithm conforms to an arbitrary PLC in three dimensions. For `SVRC`, a **collar system** (defined precisely later), is a union of balls containing all of the creases. A unique contribution of `SVRC` is the first procedure for generating a collar system with efficient worst-case runtime $O(N \log \Delta + C)$, where C is the description complexity of the collar system. This collar sizing could be easily extracted from `SVRC` as a subroutine. Previous conforming algorithms either construct collars by brute force, requiring $\Omega(NC) \in \Omega(N^2)$, or through other methods with worst case $\Omega(N^2)$. A user could, if they choose, implement the collar sizing procedure from `SVRC` and use it as a preprocess for some different conforming algorithm.

There is a larger class of input for which conforming algorithms are known, namely piecewise smooth complexes (PSCs). Here, the segments and polygons of the input may be given as smooth manifold curves and surfaces. Analogous notions of creases may be defined. Conforming algorithms must either output a PSC, or output a PLC and redefine the notion of conforming to be approximate. Boivin and Ollivier-Gooch gave a conforming algorithm for 1-PSCs without creases in two dimensions [BOG02] and suggest a method for handling creases. Cheng et al. gave the first conforming algorithm for arbitrary PSCs in three dimensions [CDR07].

[†]A note on citations: Unfortunately, in much of the meshing literature (and in other areas as well), there is often a large time gap between the first publication of results in conferences and their subsequent publication in peer-reviewed journals. The journal versions tend to be more authoritative, and I prefer to cite these wherever possible. Of course, this muddles the chronology beyond repair when I try to give due credit for first results. *Caveat Lector.*

1.3 Quality Mesh Elements

If a mesh is to simplify the geometric description, there must be some limit on the allowable complexity of mesh element. Given that an output mesh must by a 3-PLC made of features, a natural choice is to limit the class of output features allowed. I restrict my attention to algorithms that output simplicial complexes. For my purposes, a **simplicial complex** is a PLC, all of whose features are simplices.

Definition 6 (Simplex)**.** *Given a set of affinely-independent vertices V, a* **simplex** $\mathrm{Sim}(V)$ *is the convex closure of V. If $|V| = i + 1$, then $\mathrm{Sim}(V)$ is an i-simplex.*

Vertices, edges, triangles, and tetrahedra are $\{0, 1, 2, 3\}$-simplices respectively. A simplicial complex covering a a three-dimensional domain Ω is a called **tetrahedralization** of Ω. An i-simplex is called **degenerate** if any of its points lie in a common j-hyperplane for some $j < i$. There are some natural definitions for non-degenerate simplices:

Definition 7 (Circumball, Circumcenter, Circumradius, and Diametral Ball)**.** *Given a non-degenerate i-simplex S with vertices V, the* **circumball** *is the unique i-ball going through V with* **circumcenter** *c and* **circumradius** *r. For points in $\mathbb{R}^d$, if $i \leq d$, then the* **Diametral Ball** *of S is the d-Ball $B_d(c, r)$.*

1.3.1 Quality Tetrahedra

Any good meshing algorithm must provide some guarantee on the quality of the output simplices. The strictest quality bound is given by tetrahedral **aspect-ratio**.

Definition 8 (Aspect-Ratio)**.** *Given a tetrahedron T, define the* **inradius** *r_T as the radius of the largest ball contained in t. Let R_T be the circumradius of T. The* **aspect-ratio** *of t is given by R_T/r_T.*

Good quality is given by bounding the aspect-ratio from above by some constant. Aspect-ratio can be defined similarly for triangles in two dimensions. It is clear that if a tetrahedron has bounded aspect-ratio, then there is another constant bounding the aspect-ratio of its triangular faces. Bounded aspect-ratio tetrahedra have dihedral angles between adjacent triangles bounded from below, and additionally have no small angles between adjacent edges. A frequently-employed, looser condition on a tetrahedron is given by the **radius-edge** condition:

Definition 9. *Given a tetrahedron T, let $|e|$ be the length of the shortest edge of T, and let R_T be the cirumradius. The* **radius-edge** *ratio is given by $R_T/|e|$.*

Quality guarantees are given by bounding the radius-edge from above by some constant. Bounded radius-edge tetrahedra have no small angles between edges. A tetrahedron with

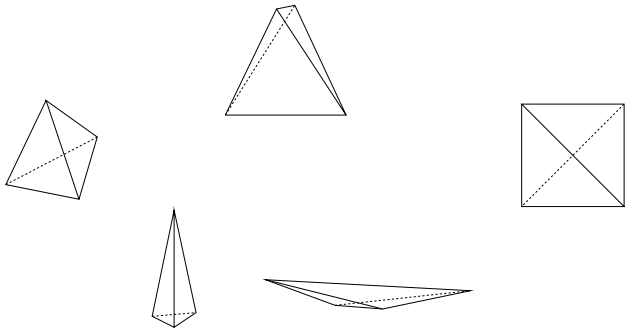

Figure 1.1: Several characteristic tetrahedra. Only the nice tetrahedron on the left has good aspect-ratio. The right most tetrahedron is a *sliver*, its triangles are nice, but the points are almost coplanar. The nice tetrahedron and the sliver are the only two with good radius-edge. The three tetrahedra in the top-left are those with no large angles.

bad radius-edge is called **skinny**. Bounded aspect-ratio implies bounded radius-edge, but not the converse. The exception to the converse is given by **slivers**, tetrahedra with good radius-edge but poor aspect-ratio. (See Figure 1.3.1). Slivers are notoriously hazardous to meshing algorithms, I discuss an approach in Section 6.2.1 that follows [Li03].

1.3.2 Delaunay and Voronoi

The most powerful tool for creating tetrahedralizations is the **Delaunay diagram** of a point set [Del34]. The Delaunay diagram is the set of simplices that are contained in empty balls.

Definition 10 (Delaunay diagram)**.** *Let M be a point-set. Define the* **Delaunay diagram** *as a set of non-degenerate simplices:*

$$\mathrm{Del}(M) := \{\mathrm{Sim}(S) \mid S \subset M \ \exists \ \textit{ball } B,\ S \subset B \textit{ and } B^\circ \cap M = \emptyset\}$$

Note that since the interior B° is disjoint from M, but $S \subset M$, then the vertices of S must be on the boundary of the ball B. If the points in M are in general position, $\mathrm{Del}(M)$ forms a tetrahedralization of the convex closure of M. `SVRC` primarily uses the Delaunay diagram to define its mesh simplices. When this is the case, a mesh M can be considered as just a set of vertices with an implicit simplical complex. This approach has slight problems with degeneracy. When five points lie on a common sphere, the Delaunay diagram as defined is not a tetrahedralization. Which subset of tetrahedra are chosen to give a tetrahedralization? `SVRC` handles this with tie-breaking rules, as in previous literature [PW04]. As a post-process,

SVRC abandons the Delaunay simplices in collars adjacent to creases, replacing these with a manually constructed tetrahedralization (see Section 3.1.2.)

What makes the Delaunay so powerful a tool is the use of its dual, the **Voronoi diagram**. The Voronoi diagram of a point set M is given by partitioning the whole domain into **Voronoi cells**, regions with a common nearest-neighbor in M. More precisely:

Definition 11 (Voronoi Polytopes)**.** *Given a point set M and non-empty $S \subset M$, define the* **Voronoi polytope** *of S with respect to M as*

$$V_M(S) := \{x \in \Omega \mid \ \forall\ m' \in M,\ \forall\ s \in S,\ |xs| \leq |xm'|\}$$

If M is in general position in $\mathbb{R}^3$, then $V_M(S) = \emptyset$ for $|S| > 4$.
For $|S| \in 1 \ldots 4$, if $V_M(S)$ is nonempty, call it a **Voronoi cell, facet, edge, or corner** *respectively. If $m, m' \in M$ with $m \neq m'$, then m and m' are Voronoi* **neighbors** *if $V_M(m) \cap V_M(m') \neq 0$. $\mathrm{Vor}(M)$ is the* **Voronoi diagram** *of M, the set of all Voronoi polytopes. $\mathrm{Vor}(M)$ covers Ω. M may be suppressed to write $V(S)$ when the context is clear.*

Given M, define the **dual** to a nonempty voronoi cell $V_M(S)$ as the simplex formed by S, so $\mathrm{Dual}(V(S)) = \mathrm{Sim}(S)$ and $\mathrm{Dual}(\mathrm{Sim}(S)) = V_M(S)$. The Voronoi cells are dual to the Delaunay vertices; facets are dual to Delaunay edges; Voronoi edges are dual to Delaunay triangles; and Voronoi corners are the circumcenters of their dual Delaunay tetrahedra. An overview of Delaunay and Voronoi and this duality is in [Ede01].

Guarantees on the quality of the Voronoi diagram can be transferred to the Delaunay diagram and back. Define the aspect-ratio of a Voronoi cell as follows:

Definition 12 (Voronoi Aspect-Ratio)**.** *Given a domain Ω, $m \in M$, define the* **inradius** *of m, r_m^M, as the radius of the largest m-centered ball contained in m's Voronoi cell, that is:*

$$r_m^M := \max\{r \in (0, \infty) \mid B(m, r) \subset V_M(m)\}$$

Symmetrically, define the **outradius** *of m, R_m^M, as the radius of the smallest m-centered ball that contains m's Voronoi cell:*

$$R_m^M := \min\{R \in (0, \infty) \mid V_M(m) \subset B(m, R)\}$$

The aspect-ratio of the Voronoi cell $V_M(m)$ is given by R_m^M / r_m^M. M may be suppressed when the context is clear.

Note that inradius, outradius, and aspect-ratio are overloaded to apply to either Voronoi cells or tetrahedra, but the context will always be clear. Voronoi cells with bad aspect-ratio will also be referred to as **skinny**.

I have only defined Voronoi aspect-ratio for bounded domains Ω, so that R_v is never infinite. A typical extension of these definitions to infinite domains is the lesser requirement

that the outradius of $V(v)$ need merely be large enough to contain the Voronoi corners and not the whole (possibly infinite) cell. In the case of an infinite domain, most of the structural results on Voronoi Diagrams (for instance Section 2.2) that would hold for all $x \in \Omega$ will still hold, but only for x in the convex closure of the point set M.

A point set M with the aspect-ratio of every Voronoi cell bounded above by some $\tau \in (1, \infty)$ is said to be **τ-Well-Spaced**. This is more restrictive than the radius-edge condition, since for any vertex, it upper bounds the ratio of the outradius of *any* adjacent tetrahedron to *any* adjacent edge. More precisely, a point set that is τ-well-spaced has Delaunay tetrahedra with radius-edge ratio bounded by $\tau/2$. The converse is non-trivial; a Delaunay diagram with tetrahedra that have radius-edge at most τ has vertices that are $(K \cdot \tau)$-well-spaced, where K is a constant depending exponentially on dimension [MTT+96].

`SVRC` operates primarily on the Voronoi diagram (hence "Voronoi Refinement"). This serves to provide cleaner analysis. Almost all of the operations in `SVRC` can be viewed as operations on the dual Delaunay without any trouble. `SVRC` attempts to generate a well-spaced point set M so that $\text{Del}(M)$ conforms to input $\mathcal{P}$. This is possible nearly everywhere in the domain, except near sharp creases. To fit conforming tetrahedra in a crease, some tetrahedra must have small edge angles, making well-spaced vertices impossible. Adjacent to the creases, `SVRC` still makes some guarantee about the output tetrahedra.

The loosest guarantee on the quality of a tetrahedral element is given by the **No-Large-Angle** (NLA) condition, an upper bound on the angle between any adjacent triangles[‡]. Simplices output from `SVRC` adjacent to creases have an NLA guarantee. The problem of NLA meshing in general is theoretically quite interesting. There are several algorithms for NLA meshing in two dimensions [Tan96, MPS07]. In three dimensions the problem is not as well-understood, there are some results for point-set inputs [BEG94, MPS08].

For the body of this thesis, a **τ-quality mesh** will simply refer to a τ-well-spaced set of vertices. A τ-quality mesh is conforming to $\mathcal{P}$ if $\text{Del}(M)$ conforms to $\mathcal{P}$. Very strong structural statements can be made about quality meshes, most notably that the Delaunay diagram has bounded degree (Section 2.2.2).

1.3.3 Computational Needs

Dithering about quality guarantees on mesh elements is moot without reference to the needs of downstream computations. Numerical methods are the chief application for meshes, and such methods have two basic needs related to element quality: correctness (guaranteed convergence to a solution), and efficiency (rate of convergence). Without delving into numerical analysis, we can summarize a few necessities.

There are two classical results in the area. The first are the so called Ciarlet-Raviart [Cia78] results in interpolation theory showing that bounded aspect-ratio tetrahedra are

[‡]This condition is not strictly loosest, as slivers satisfy good radius-edge but not NLA.

sufficient for a wide variety of numerical methods. The second is Babŭska-Asiz [BA76], showing that NLA tetrahedra are necessary and sufficient for correctness of most numerical methods. Many simple downstream numerical methods will prove correct for NLA elements, but the efficiency will be damaged by the worst aspect-ratio tetrahedron. Some state of the art methods allow for efficiency only related to the largest dihedral angles [BHV04, MPV05]. Some control-volume methods can be run effectively with only a good-aspect ratio Voronoi diagram [MTT+96].

1.4 Mesh Sizing

Also crucial to downstream numerical applications will be the size of the output mesh. In this section, *size* is cleverly overloaded to refer to both the number of output elements as well as their geometric size, since the two are intimately related. Elements must be few in number to make future computations tractable, and must be small enough to approximate the geometry and physics with accurate resolution.

Creating mesh elements small enough to capture the physics is a rather uninteresting problem in mesh generation. Once a user has a quality conforming mesh, any number of naive refinement methods for adding more vertices will suffice to create a mesh with smaller elements for resolving physics. A typical method for PLCs without creases is given in [BOG01]. The makes the more interesting question that of asking for the *minimum* number of elements required to obtain a quality conforming mesh in the first place. This is equivalent to asking for the largest elements possible, and the answer is given by the *local feature size* as defined by Ruppert [Rup95][§].

Definition 13 (Local Feature Size f)**.** *Given a point set M, the local feature size $f_M : \mathbb{R}^3 \to (0, \infty)$ is given by the distance to the second nearest neighbor:*

$$f_M(x) := \operatorname*{argmin}_{r \in (0,\infty)} [|B(x, r) \cap M| \geq 2]$$

This extends to any set $\mathcal{X}$ of features as the shortest distance to two disjoint features:

$$f_{\mathcal{X}}(x) := \operatorname*{argmin}_{r \in (0,\infty)} [\exists\ F, G \in |B(x, r) \cap \mathcal{X}|,\ F \cap G = \emptyset]$$

Observe that f is bounded away from zero. The seminal result of [Rup95] was to use this sizing as a lower-bound on the number of triangles in a quality mesh of a PSLG without creases. This was later extended to PLCs without creases in three dimensions in [MV00].

[§]I will eschew the terminology "local feature size" and "lfs" as much as possible as it is unfortunately overloaded in the computational geometry research community. Other works on meshing define variants as the "mesh feature size", "current feature size", "lfs_1", "lfs_0", etc. Local feature size has another competing and unfortunately similar definition relative to manifolds and the medial axis. The ambiguous overloading from smooth surfaces to PLCs is most likely attributable to Ruppert's original paper.

Theorem 1 (Ruppert Lower Bound)**.** *In three dimensions, for any PLC $\mathcal{P}$ without creases, any mesh M τ-aspect-ratio tetrahedra must have:*

$$|M| \gtrsim \int_{\Omega} \frac{1}{f_{\mathcal{P}}^3}$$

The bound is realized when tetrahedra in a mesh M have circumradii proportional to $f_{\mathcal{P}}$, so then $f_M \sim f_{\mathcal{P}}$. Then the volume of an element is roughly f^3, making the density of vertices $1/f^3$. Ruppert's algorithm and many subsequent algorithms meet this bound to within a constant, showing that the lower bound is tight. These algorithms are called *size-optimal*, since they output an $O(1)$-approximation to the optimal output in terms of size [Rup95, She98]. The algorithm `SVRC` is based on the earlier Sparse Voronoi Refinement (`SVR`) alogrithm [HMP06] that handles PLCs without creases. `SVR` is a size-optimal algorithm, and `SVRC` inherits this property if the input has no creases. This is apparent from the analysis in Chapter 4 if one assumes the absence of creases.

The value of $\int 1/f^3$ can be unbounded relative to the input size N. It is particularly related to the spread Δ. For general PLCs, the only upper bound is Δ^3. However, for point set inputs N, the integral can be easily bounded above by $|N| \log \Delta$. Of particular pertinance is that in many specific cases, this integral can be upper bounded by $|N|$.

Cases where $|M| \lesssim |N|$ occur when the input is essentially a quality mesh of some sub domain (possibly in lower dimension). In this case, the additional vertices of M serve only to fill out the domain and provide supporting structure to N. I codify this new result as the "Scaffold-Sizing Theorem" of Chapter 4, and make use of it to prove efficiency bounds on `SVRC` in Chapter 5.

Unforuntately, when handling PLCs with creases, the sizing analysis is not so clear. The output mesh must be allowed some skinny elements near creases, but if skinny elements are allowed throughout the domain then all bets are off; the lower bound vanishes entirely and size-optimality is a withered concept.

To make some statement about mesh sizing, I employ the following definition of a sizing function g following [CDR07]¶.

Definition 14 (Sizing g)**.** *Given a set of features $\mathcal{X}$, the gap-size $g_{\mathcal{X}} : (\mathbb{R}^3 \to \mathbb{R})$ is defined at x as the shortest distance to two features, one of which does not contain x:*

$$g_{\mathcal{X}}(x) := \operatorname*{argmin}_{r \in (0,\infty)} \left[|B(x,r) \cap \mathcal{X}| \geq 2 \text{ and } \exists\ F \in B(x,r) \cap \mathcal{X},\ x \notin F \right]$$

From the definitions it is clear that $g < f$ in all circumstances. Observe that when $\mathcal{P}$ is a point set, $f = g$. While f is fairly well behaved (Chapter 2), g is highly discontinuous as x moves from feature to feature. See Figure 1.4.

¶Cheng et al. refer to this function as the "gap-size" which is an unfortunately overloaded name. This function has other names as well [Üng04, RW08]. I will later employ a competing definition of gap-size from[Mil04].

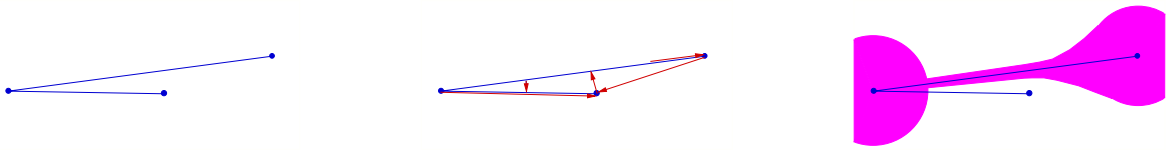

Figure 1.2: On the left, an PLC consisting of three vertices and two segments meeting at a crease. Center, the arrows indicate the measurement of $g(x)$ at their tails. Along the segment, g will go to zero as x approaches the crease point, but then g is large again at the crease vertex. At right, suppose the top segment is a crease between some unseen facets, then the collar region is shown.

Despite its non-smooth behavior, the utility of g becomes apparent upon considering the definition of following domain: the **collar region** defined as

Definition 15 (ε-Collar-Region)**.** *Let $0 < \varepsilon < 1/3$. Suppose $\mathcal{P}$ is a PLC and let C be the union of the creases of $\mathcal{P}$, then the* **ε-collar-region** *is given by*

$$\hat{\mathcal{C}} := \bigcup_{x \in C} B(x, \varepsilon g(x))$$

This region is nice in that locally (within one of the balls), it will only ever intersect a crease and the features containing that crease, and it is roughly the largest region that will isolate the creases. The choice of $1/3$ as a maximum for ε is slightly arbitrary. It must be a constant factor smaller than $1/2$ to isolate creases and to prevent non-local interaction between creases. This makes it attractive to designate as a special region where creases can be treated separately. Accordingly, the collar system employed by `SVRC`(following previous algorithms) will be an approximation to this collar region.

Since it is useful to consider $\mathcal{P}$ with the collar region removed, consider the following:

Definition 16 (ε-Clipped-Complex)**.** *Consider PLC $\mathcal{P}$ and its ε-collar-region $\mathcal{R}$. The* **ε-clipped-complex** *$\mathcal{P}'$ is the set of features with portions removed inside.*

$$\mathcal{P}_{\mathcal{R}} := \{F - \mathcal{R} \mid F \in \mathcal{P}\}$$

Note that the clipped complex is not in general a PLC.

Then consider the clipping the collar region to form $\mathcal{P}_{\hat{\mathcal{C}}}$. Although it is not a PLC, the sizing functions f and g are still well defined with regards to $\mathcal{P}_{\hat{\mathcal{C}}}$. Since `SVRC` will generate well-spaced elements in the exterior of the collar region $(\Omega - \hat{\mathcal{C}})$, it makes some sense to guarantee that the radii of these elements is proportional to $f_{\mathcal{P}_{\hat{\mathcal{C}}}}$.

This is "optimal" in some sense with respect to $\mathcal{P}_{\hat{\mathcal{C}}}$, but not necessarily optimal with respect to $\mathcal{P}$. Lower bounds for meshes of PLCs with creases remains an open problem. The

upper-bound on SVRCcould be shown for other conforming algorithms for PLCs with creases, asymptotically, all of the collar-based algorithms generate asymptotically similarly output [CP06, PW05, RW08]. There are other algorithms based on "constrained Delaunay" that allow skinny elements in a larger region than simply adjacent to creases. Since they allow worse elements, these typically achieve far fewer elements in practice[SG05], but no better sizing bounds have been shown.

1.5 Efficient Algorithms

Conforming, element quality, and sizing are all issues of correctness for meshing algorithms, but the principle contribution of this thesis is in the last subproblem, algorithm efficiency. The question of runtime complexity for meshing algorithms is an ongoing area of research. There are two primary caveats in this area, the complexity of the Delaunay diagram of a set of points, and the costs of point location.

1.5.1 Delaunay Complexity

The traditional paradigm for Delaunay-based mesh generation algorithms has been to begin with the Delaunay diagram of the vertices of the input PLC, and then add new vertices to the mesh toward the goals of conforming and element quality, all the while updating the Delaunay diagram [Che89b, Rup95, She97a, PW04]. This approach has a fatal flaw in the realm of worst-case runtime. The number of edges in the Delaunay diagram of N vertices in three dimensions can have complexity in $\Omega(N^2)$, which is realized by very simple pathologies. For many examples, there is a quality output mesh with M vertices and $M \in O(N)$. This quality mesh will always have complexity in $O(M)$[MTTW99], so any algorithm with a cost $\Omega(N^2)$ is ruined for worst-case efficiency. This is a problem for all the existing algorithms that handle creases.

To avoid paying the unacceptable quadratic extra cost, this caveat was overcome for Delaunay refinement algorithms in [HMP06] using the "Sparse Refinement" paradigm. As in previous algorithms, new vertices are inserted to help reach the goals of element quality and conforming to the input. Traditional Delaunay refinment algorithms first find a conforming mesh, and then achieve quality. Sparse refinement algorithms put the cart before the horse; always keeping a quality mesh and then working towards a conforming mesh.

This thesis and the SVRC algorithm are the first to extend the sparse refinement paradigm to handle PLCs with creases. SVRC is the first meshing algorithm for PLCs with creases that has an efficient worst-case runtime. Following sparse refinement, the SVRC algorithm iteratively maintains a Delaunay diagram, but begins with a much coarser diagram not conforming to any of the input. Then, SVRC maintains a strong invariant; the current complexity (number of edges, triangles, tetrahedra) is forced to be only *linear* in the current number of vertices. This invariant is achieved by maintaining a quality mesh at all times.

Even though the quality may at times be much worse than the end product, no element is ever arbitrarily skinny. This quality invariant gives a degree-bound (Section 2.2.2), ensuring the complexity is linear in the number of vertices.

Another class of meshing algorithms are those based on quadtree or octree refinement methods [BEG94, MV00]. These can also be viewed as sparse refinement algorithms. They do not suffer the Delaunay complexity problem, however they suffer when handling the second major cost in meshing: point location. None of these algorithms have non-trivial runtime bounds for handling PLCs.

1.5.2 Point Location Costs

The classic problem of point location in computational geometry is as follows: given a partition of the domain and a point, find the element of the partition containing the point. This problem arises in mesh generation when an algorithm does not conform to the input. When a meshing algorithm wishes to add some new input vertex to an existing mesh, the algorithm must locate a set of current elements to modify. Similarly, a meshing algorithm must track the locations of all the segments and facets that it must conform to.

The process of point location is analagous to the simple process of search in an ordered set, and so expect the time to locate a point in N elements should be around $O(\log N)$. In quality meshes, however, element radii will be graded, so that adjacent elements only differ in size by a constant factor. Accordingly, search structures often have depth $O(\log \Delta)$, since Δ governs the ratio of largest to smallest elements. The point location structure in `SVRC` will have this depth. For most practical inputs, $\Delta \in \mathit{Poly}(N)$, so the location costs become $O(\log N)$ per location.

Tetrahedralization can be reduced to sorting, by setting up gadgets so that the sorted input appears as an easily recoverable path in the output mesh [PS85]. For an input of size N, in a comparison-based model, this gives a lower bound of $\Omega(N \log N)$. If a meshing algorithm produces an output of size N, then the lower bound becomes the output-sensitive $\Omega(N \log N + M)$.

1.5.3 Related

The quadtree based methods achieve $O(N \log \Delta + M)$ when the input is a point set [BEG94, MV00]. Unfortunately, these algorithms have trouble with location costs when the input is a PLC, so the only analyzed bound is a trivial $O(M^3)$. A notable algorithm for point sets is the fast offcenter algorithm described in [HPÜ05], which uses a hybrid of quadtree and Delaunay refinement techniques. Spielman et al. gave an algorithm for PSLGs without creases that runs in $O(\log^2 \Delta)$ parallel time, but without a work bound [STÜ07].

Miller provided the first sub-quadratic time bound for PSLGS without creases in 2D, achieving $O((N \log \Gamma + M) \log M)$, where Γ is a localized version of Δ (in particular, $\Gamma \leq$

Δ)[Mil04][‖]. The original SVR gave the first efficient algorithm for PLCs without creases of $O(N \log \Delta + M)$[HMP06]. This was extended to a parallel version with depth $O(\log \Delta)$ and the same work bound [HMP07]. SVRC is the first algorithm to achieve this bound for general PLCs.

When run on a PLC $\mathcal{P}$ with size N and outputting $\mathcal{T}$ with size M, SVRC has worst case runtime bounded by $O(N \log \Delta + M)$. This worst case bound is a vast improvement over any previous algorithms for PLCs with creases. For most practical inputs, $\Delta \in Poly(N)$, so this bound is becomes optimal $O(N \log N + M)$.

SVRC also has optimal space usage $O(M)$. This is not generally difficult to achieve for any meshing algorithm that avoids the Delaunay complexity.

[‖]The function Γ also plays a role in the runtime analysis of SVRC. It is defined in Section 2.2.3.

Chapter 2

Preliminaries

In this chapter, I will cover useful background results in the area of mesh generation that will be crucial to the analysis of SVRC. Nothing in this chapter is particularly new, although in many cases I have rephrased well known results and definitions to suit my needs.

Section 2.1 covers simple, standard, and straightforward results. Variants on these lemmas appear in nearly all research on Delaunay-based mesh generation algorithms. The lemmas in Section 2.1.1, particularly Lemmas 1 and 3 are heavily used throughout the thesis. The Proximal Packing Lemma in Section 2.1.2 is a fairly straightforward volume-packing result. This result is heavily used in the runtime analysis of Chapter 5. Accordingly, most previous work on runtime-efficient meshing algorithms utilize similar results [STÜ07, Mil04, HMP06]. I provide an elegant formulation of this lemma that is particularly well-suited for future reuse.

Section 2.2 discusses structural properties of quality Voronoi diagrams, in particular the degree-bound [MTTW99] that is the heart of the sparse refinement paradigm for efficient meshing algorithms. A wonderfully thorough exposition containing most of the results in this area appears in [Tal97]. For analyzing SVRC, I develop these results on the essentially the same path as in [HMP06].

The Well-Pacing Theorem is presented with some other results on mesh size analysis in Section 2.3. These results are needed to establishing the Scaffold-Sizing Theorem in Section 4.2.

The last section presents some background results in surface reconstruction that will be used by SVRC for the implicit construction of the collar system. The full strength of this groundwork is used for constructing meshes to approximate smooth manifolds, see [Dey06] for a treatise. SVRC protects the collar region by a union of spheres, greatly simplified lemmas suffice. I principally use the results of this section to show the correctness of SVRC, with regards to conforming to the PLC near the creases and element quality within the collar system.

2.1 Standard Results

2.1.1 Sizing Functions and Voronoi Diagrams

It is of some use to define a function simpler than f and g, the nearest-neighbor function n.

Definition 17 (Nearest-Neighbor Distance)**.** *Consider a PLC $\mathcal{P}$ and $x \in \Omega$ be given. Define the* **nearest-neighbor distance***:*

$$n_{\mathcal{P}}(x) := \operatorname*{argmin}_{r \in [0,\infty)} [B(x,r) \cap \mathcal{P} \neq \emptyset]$$

$\mathcal{P}$ may be suppressed.

A simple observation is that $n_{\mathcal{P}} < f_{\mathcal{P}}$ for all $\mathcal{P}$. If M is a point set with $m \in M$, then $x \in V_M(m)$ implies $n_M(x) = |xm|$, in particular $n_M(m) = 0$. If m is removed to get $M' = M - \{m\}$, then $f_M(m) = n_{M'}(m)$.

Consider a PLC $\mathcal{P}$ and the sizing functions $f_{\mathcal{P}}$ and $g_{\mathcal{P}}$. It is useful to make some standard claims on the smoothness of f and g [Rup95, CDR07].

Definition 18 (Lipschitz)**.** *A function h is α-Lipschitz if:*

$$\forall\ x, y,\ h(x) \leq h(y) + \alpha|xy|$$

A function is called **Lipschitz** *if it is 1-Lipschitz.*

Lemma 1 (f is Lipschitz)**.** *$f_{\mathcal{P}}$ and $n_{\mathcal{P}}$ are Lipschitz for any $\mathcal{P}$.*

Proof. Let x, y and consider $B := B(x, n(y) + |xy|)$. Clearly $B(y, n(y)) \subset B$, so it follows that $B(y, n(y)) \cap \mathcal{P} \subset B \cap \mathcal{P}$. Since by definition of n, $B(y, n(y)) \cap \mathcal{P} \neq \emptyset$, then the superset $B \cap \mathcal{P} \neq \emptyset$, so $n(x) \leq |xy| + n(y)$. The same argument holds for f, replacing "$\neq \emptyset$" everywhere with "intersects two disjoint features of $\mathcal{P}$". □

Lemma 1 is the bread and butter of meshing analysis, and will be invoked an unimaginable number of times in this thesis, often without reference. The function g is not as nice as f, but it is smooth in a localized sense. On restricting the domain to the interior of a single feature, then g is indeed smooth:

Lemma 2 (g is Lipschitz on a feature)**.** *Let F be a feature in $\mathcal{P}$, then $g_{\mathcal{P}}(x)$ restricted to x in the interior of F is Lipschitz.*

Proof. Since the domain is restricted the interior of F, the definition of g is reduced to the following, Let $\mathcal{P}' := \mathcal{P} - \{G \in \mathcal{P} \mid F \not\subset G\}$ be the set of features that do not contain F, so that g is just the nearest neighbor from $\mathcal{P}'$. In the restricted domain $g_{\mathcal{P}} = n_{\mathcal{P}'}$, so g inherits the Lipschitz property from n. □

The following lemma codifies some simple but useful facts on f and n in relation to Voronoi diagrams. Lemma 3 will be invoked countless times in later analysis.

Lemma 3. *Consider any point set M and any $v, w \in M$. Consider any $x \in V_M(v)$, then:*

$$n_M(x) = |xv| \leq f_M(x)$$

$$r_v = \frac{f_M(v)}{2} \leq \frac{|vw|}{2}$$

$$r_v \leq f_M(x) \leq 3R_v$$

Proof. The first fact is trivial from definitions.

For the second fact, consider that by the definition of f, $f_M(v)$ is the distance to the nearest neighbor of v in $M - \{v\}$, so the upper bound is trivial. To see that $r_v = f(v)/2$, let u be a nearest neighbor of v. Consider the ball $B(v, |uv|/2)$. This ball is completely contained in $V(v)$, otherwise there would be some vertex closer to v than u. But any larger ball intersects the interior of $V(u)$ so is not contained in $V(v)$, thus $r_v = |uv|/2 = f(v)/2$.

The upper bound on f in the third fact is not difficult. By Lipschitz and the second fact:

$$f(x) \leq |xv| + f(v) = |xv| + 2r_v \leq R_v + 2r_v \leq 3R_v$$

To see the lower bound on f in the third fact, consider two cases. First, if $r_v \leq |xv|$, then immediately $r_v \leq f(x)$ by the first fact. Second, if $|xv| \leq r_v$, then by Lipschitz on f and the second fact,

$$f(x) \geq f(v) - |xv| = 2r_v - |xv| \geq 2r_v - r_v = r_v$$

□

2.1.2 Proximal Packing Lemma

In this section, I describe a straightforward packing lemma that will be used several times. This lemma bounds the size of a sequence of disjoint events around a central origin, and will be main tool for proving bounds of the form $O(\log \Delta)$.

Definition 19 (Proximal Event Sequence)**.** *Define an* **event** *as a pair consisting of a ball and a vertex, denoted $\langle B(u, r), v\rangle$. An* **event sequence** *U around v is an ordered set of events $\langle B(u_i, r_i), v\rangle \in U$ with the following interior disjointness property:*

$$\forall\ i,\ \forall\ j < i,\ u_j \notin B^\circ(u_i, r_i)$$

so that each event point u has a ball around it whose interior is empty of previous event points.
An event sequence is a **ν-proximal event sequence** *if there exists $\nu \in (0, \infty)$ such that:*

$$\forall\ i,\ r_i \geq \nu |u_i v|$$

I now prove a bound on the size of a proximal event sequence around a vertex. Statements of this form appear in many works, including [HMP06]. The framework here is written to suit the needs of this thesis.

Lemma 4 (Proximal Event Packing)**.** *Suppose U is a ν-proximal event sequence around v with distances from v bounded by $r \leq |u_i v| < R$, then:*

$$|U| \lesssim \log(R/r)$$

When $R/r \lesssim \Delta$, this gives:

$$|U| \lesssim \log \Delta$$

If all the events are at roughly the same distance d, i.e. $|u_i v| \sim d$, then $R/r \lesssim 1$ so that:

$$|U| \lesssim 1$$

Proof. The proof is by packing events into a sequence a of size-doubling annuli around v. Define the k^{th} annulus around v as the difference of two balls given by $A_k := B(v, 2^{k+1}r) - B(v, 2^k r)$. Partition the events into sets $U_k := \{\langle B(u_i, r_i), v\rangle \mid \ u_i \in A_k\}$.

All of the U_k are empty when $k > \log(R/r)$. To finish the lemma, I claim that $U_k \lesssim 1$ for all the non-empty annuli.

Consider the set U_k for some fixed k, and consider any pair of events $i < j \in U_k$. By disjointness of the sequence, $u_i \notin B^\circ(u_j, r_j)$, so there are two smaller balls $B(u_i, r_j/2)$ and $B(u_j, r_j/2)$ that are disjoint. Letting r_k be the smallest radius of any event in U_k, there is a family of disjoint balls $\mathcal{B}_k := \cup_i B(u_i, r_k/2)$.

The small radius r_k was the radius of some event j so it follows directly from assumptions that $r_k \geq \nu |u_j v| \geq \nu 2^k r$. If $r_k > 2^k r$, then shrink the balls in $\mathcal{B}_k$ further by reducing $r_k := 2^k r$. This now guarantees that $r_k \sim 2^k r$. Then each ball in $\mathcal{B}_k$ has $Vol(B) \sim 2^{3k} r^3$. Since the balls are disjoint, the volume of $\mathcal{B}_k$ is found by summing to get $Vol(\mathcal{B}_k) \sim |U_k| 2^{3k} r^3$, with constant depending on ν.

This centers are in the annulus, but the entire balls may not be, so consider a fattening of the annulus A_k by an additive factor of r_k, to define:

$$\bar{A}_k := B(v, 2^{k+1}r + r_k) - B(v, \max(0, 2^{\mathrm{k}}\mathrm{r} - \mathrm{r_k}))$$

so that $\mathcal{B}_k \subset \bar{A}_k$. Since $r_k \sim r$, this fattening does not much change the size of the annulus, so it follows from simple geometry that $\bar{A}_k$ has volume $Vol(\bar{A}_k) \sim 2^{3k} r^3$.

All of the balls in B_k are completely contained in $\bar{A}_k$, so $Vol(B_k) \leq Vol(\bar{A}_k)$. Summarizing:

$$Vol(\bar{A}_k) \sim 2^{3k} r^3 \sim \frac{Vol(B_k)}{|U_k|} \leq \frac{Vol(\bar{A}_k)}{|U_k|}$$

thus $|U_k| \lesssim 1$, with constant depending only on ν. □

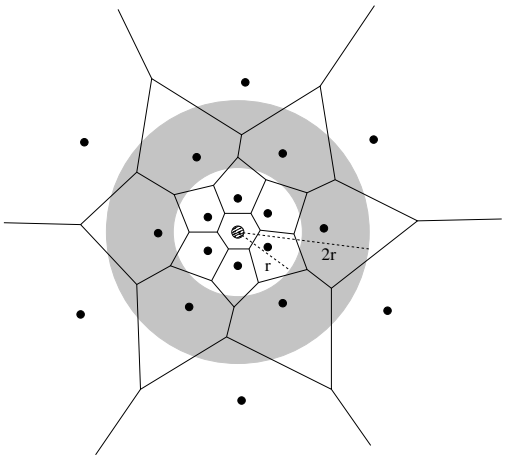

Figure 2.1: An illustration showing the intuition behind Lemma 4 and its later uses. A quality Voronoi diagram packs vertices around the small central feature. There are only a constant number of vertices in each radius-doubling annulus (shown in gray).

In Chapter 5, this lemma is utilized to show that all of the nonlinear point-location work can be amortized as one proximal event sequences around each input feature. The analysis of `SVRC` also uses this lemma to bound the work used to calculate the size of the collar region.

2.2 Structural Properties of Quality Voronoi Diagrams

In this section, I overview key structural properties of well-spaced points that will be important to the runtime analysis of `SVRC` in Chapter 5. I will first define the notion of a **gap ball**, a notion originally defined for mesh coarsening [MTT99, Tal97]. A gap ball is simply a ball whose interior is empty of vertices:

Definition 20 (Gap-Ball)**.** *Let M be a point set in Ω. A* **gap-ball** *$B(c, r)$ of M is any ball such that $c \in \Omega$ and the interior of B is empty of vertices from M, $B^\circ \cap M = \emptyset$.*

2.2.1 Gap Ball Sizing Lemma

The reason for defining gap balls is that in a quality mesh, the sizing f is strongly tied to the size of gap balls. The goal of this subsection is to show the Gap Ball Sizing Lemma. This powerful result states that in a quality mesh M, the sizing f_M at any point in a gap-ball of M is bounded below by a constant times the ball's radius. This lemma and proof is appear in slightly different language [HMP06] and absolutely would not have been possible without my co-authors on that work. I have cleaned up the proof here to avoid a messy use of Cartesian coordinates that previously appeared. A similar result with a substantially

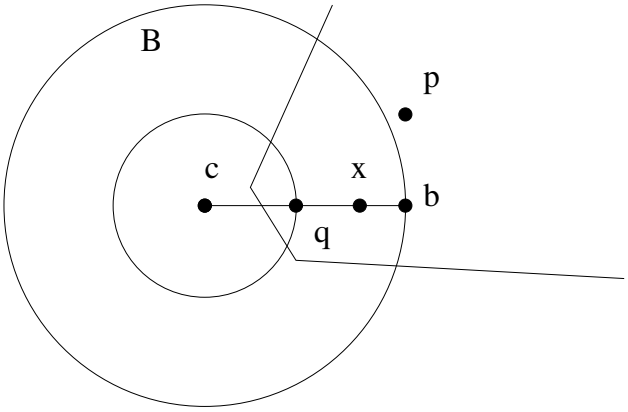

Figure 2.2: A Gap Ball B such at $V(p)$ contains q and b.

different proof appears in [MTTW99]. The proof approach taken here has constants that are notably independent of dimension.

Lemma 5 (Gap Ball Sizing)**.** *Suppose that M is a τ-quality Voronoi diagram, and suppose that $B(c, r)$ is a gap-ball of M. If $x \in B$ then*

$$f_M(x) \gtrsim r \tag{2.2.1}$$

with constant depending only on τ.

Proof. Let $B(c, r)$ be a gap ball for M as in the hypothesis. By scaling it will suffice to prove the lemma for $r = 1$ and show that f is bounded below by a constant.

Let $x \in B$ be given. The first simple observation is that $f(x) \geq 1 - |xc|$. If $|xc| \leq 1/2$ then $f(x) \geq 1/2$ and the proof is done. Subsequently assume that $|xc| \geq 1/2$.

Consider the ray $\overrightarrow{cx}$. Define q and b as the points on $\overrightarrow{cx}$ at a distance $1/2$ and 1 from c. Let $p \in M$ so that $q \in V(p)$. See Figure 2.2.

By Lemma 3, it holds that $|qp| \leq R_p$. There are two cases depending on whether or not $b \in V(p)$.

Case 1: Suppose that $b \in V(p)$. Since both q and b are in $V(p)$ it follows that $x \in V(p)$ and by Lemma 3, $f(x) \geq r_p$. Since B is a gap ball, p cannot be interior so $1/2 \leq |qp|$. Using τ-quality, it then follows:

$$f(x) \geq r_p \geq R_p/\tau \geq \frac{1}{2\tau} \tag{2.2.2}$$

Case 2: Suppose that $b \notin V(p)$. In this case, the goal is to get a stronger lower bound on $|qp|$. Since $b \notin V(p)$, then $|bp| > r_p$. The following chain of inequalities holds:

$$|qp| \leq R_p \leq \tau r_p \leq \tau |bp| \tag{2.2.3}$$

The remaining proof requires some light linear algebra. Without loss, assume that c is the origin. Observe $b = 2q$, $b^T b = 1$, $q^T q = 1/4$, and $pTp \geq 1$. Consider further:

$$|qp|^2 = (p-q)^T(p-q) = p^T p - 2p^T q + q^T q = p^T p - 2p^T q + 1/4 \tag{2.2.4}$$

$$|bp|^2 = p^T p - 2p^T b + b^T b = p^T p - 4p^t q + 1 \tag{2.2.5}$$

Squaring (2.2.3) and plugging in (2.2.4) and (2.2.5):

$$p^T p - 2p^T q + 1/4 \leq \tau^2(p^T p - 4p^t q + 1)$$

Since $\tau > 1$, cancel the $p^T p$ term and re-arrange:

$$(4\tau^2 - 2)p^T q \leq \tau^2 - 1/4 \tag{2.2.6}$$

$$p^T q \leq \frac{1}{2}\left(1 - \frac{1}{4(2\tau^2 - 1)}\right) \tag{2.2.7}$$

Back substituting (2.2.7) into (2.2.4) and using $p^T p \geq 1$, obtain:

$$|pq|^2 \geq 5/4 - \left(1 - \frac{1}{4(2\tau^2 - 1)}\right) = 1/4\left(1 + \frac{1}{2\tau^2 - 1}\right) \tag{2.2.8}$$

Define $\delta := |pq| - 1/2$. Consider then that $\delta^2 + \delta + 1/4 = |pq|^2$ and substitute into (2.2.8) to obtain:

$$\delta^2 + \delta \geq \frac{1}{4(2\tau^2 - 1)} \tag{2.2.9}$$

Use the quadratic formula to find that:

$$\delta \geq \frac{-1 + \sqrt{1 + \frac{1}{2\tau^2 - 1}}}{2} \tag{2.2.10}$$

Since $\tau > 1$, δ is positive and bounded away from zero by some $\delta_\tau > 0$ so $|pq| \geq 1/2 + \delta_\tau$. By Lipschitz then $f(x) \geq f(p) - |xp| \geq 1/2 + \delta_\tau - |xp| \geq \delta_\tau$ and the lemma is proved. □

2.2.2 Degree Bound Theorem

The lemmas just presented allow a generalization of the bounded-ply theorem of Miller *et al.* [MTTW99] stating that any gap ball intersects only a constant number of Voronoi cells. This leads directly to the degree-bound.

Lemma 6 (Bounded-Ply)**.** *Consider a τ-quality mesh M on domain Ω take any gap ball $B := B(c, r)$ with $c \in \Omega$ Then the number of Voronoi cells intersecting B is bounded by a constant depending only on τ, that is:*

$$B^\circ \cap M = \emptyset \longrightarrow |\{v \in M \mid \ V_M(v) \cap B \neq \emptyset\}| \lesssim 1$$

Proof. Let any gap ball $B := B(c, r)$ be given. There is always a maximally empty ball $B' \supset B$ with two vertices of M on its surface, so without loss, assume B has two vertices of M on its surface. Thus $f(c) = r$, and for any $x \in B$, by Lipschitz on f it follows that $f(x) \leq |xc| + f(c) \leq 2r$.

Let $M' \subset M$ be the Voronoi cells that intersect B, i.e. $M' := \{v \in M \mid V_M(v) \cap B \neq \emptyset\}$, so the goal is to show that $M' \lesssim 1$.

Consider any $v \in M'$, then there is some point $x \in B \cap V(v)$. By Lemma 3 it holds that $|xv| \leq f(x)$. The vertex v must be relatively close. The distance $|vc|$ is bounded by:

$$|vc| \leq |xc| + |xv| \leq r + f(x) \leq 3r \tag{2.2.11}$$

The Voronoi cell $V(v)$ must also be relatively large. Lemma 3 gives $f(x) \leq 3R_v$. By Lemma 5 (Gap-Ball Sizing) it holds that $f(x) \gtrsim r$, thus:

$$|vc| \leq |xc| + |xv| \leq r + R_v \lesssim f(x) + R_v \leq 4R_v \leq 4\tau r_v \lesssim r_v \tag{2.2.12}$$

The set of balls $B(v, r_v)$ given by every choice of $v \in M'$ are all disjoint, so view this set as a sequence of events around c. Equation (2.2.12) shows they are proximal events. Equation (2.2.11) shows that every event is at roughly the same distance $|cv| \sim r$. Thus $|M'| \lesssim 1$ by Proximal Packing Lemma 4. □

Lemma 7 (Worsened quality)**.** *Suppose M is a τ-quality-mesh. Let $M' = M - S$ for some small subset $S \subset M$. Then M' is a τ'-quality-mesh where τ' depends on $|S|$ and τ.*

Proof. The general case is by induction on $|S|$, proving the lemma for $|S| = 1$ suffices.

Denote the singleton vertex to be removed as s. Consider any $v \in M$. The goal is to bound change in v's aspect ratio R_v/r_v. Since $M' \subset M$, it must be the case that $V_M(v) \subset V_{M'}(v)$, so the inradius of v can only increase $r_v^{M'} \geq r_v^M$ which would improve the aspect ratio.

The outradius of v may increase, worsening the aspect ratio, but this is controlled. Any new point x in $V_{M'}(v) - V_M(v)$ must have come from $V_M(s)$. Since Voronoi cells are always convex, this implies that v and s were neighbors in M, so $|vs| \leq 2R_v^M$. Furthermore, x must have had v as its old second nearest neighbor, so $f_M(x) = |xv|$. But $f_M(x) \leq 2R_s^M$, thus $|xv| \leq R_s^M \leq \tau r_s^M \leq \tau |vs|/2 \leq \tau R_v^M$. Thus $R_v^{M'} \leq \tau R_v^M$, so the aspect-ratio of v's Voronoi cell is only worsened by a factor of τ.

By induction this yields a total worsening of $\tau^{|S|}$. □

Theorem 2 (Degree Bound)**.** *If M is a τ-quality mesh, then* $\mathrm{Del}(M)$ *has maximum degree $D \lesssim 1$.*

Proof. The proof follows immediately from the previous two lemmas. Remove v and what remains is still a quality mesh. Put down a large empty ball in the space v used to occupy.

This can only intersect a constant number of the new Voronoi cells, but these are a superset of the old neighbors of v, giving an upper bound. Formally:

Consider any vertex $v \in M$, and consider $M' := M - \{v\}$. Consider the ball $B := B(v, R_v^M)$, so $V_M(v) \subset B$. Define U as the Voronoi neighbors of v. Let $u \in U$, then:

$$\emptyset \neq V_M(u) \cap V_M(v) \subset V_M(u) \cap B \subset V_{M'}(u) \cap B$$

so u's new Voronoi cell intersects B. By Lemma 7, M' is a quality mesh. Furthermore, B is disjoint from M', so by Lemma 6, $|U| \lesssim 1$. By duality, the degree of v in $\mathrm{Del}(M)$ is exactly the same as number of Voronoi neighbors $|U|$. □

2.2.3 Grading Lemmas

I define a yet another sizing function relative to a point set M following previous work in [MTTW99, Tal97]. The utility of the gap-size is that it allows the definition of **grading**, a metric on the local quality of the mesh anywhere in the domain. Previous metrics were limited to describing quality at a vertex or on a whole element. This notion and nearly-equivalent notions for grading are essential to meshing runtime analysis and have been used before [MTT99, Tal97, Mil04, STÜ07, HPÜ05].

Definition 21 (Gap-Size)**.** *Let M be a point set in Ω. Let $x \in \Omega$. The* **gap size** *$G_M(x)$ is the radius of the largest gap-ball B of M such that x lies on the surface of the ball ∂B. M may be suppressed.*

It is worth noting that the gap size is a monotone decreasing function as vertices are added to a mesh M, since the empty interior condition on gap balls only becomes more difficult to satisfy.

Definition 22 (Grading)**.** *Let a mesh M and a point $x \in \Omega$ be given. Define the* **grading** *of M at x as:*

$$\Gamma_M(x) := \frac{G_M(x)}{f_M(x)}$$

M may be suppressed.

Lastly in this section, I present two converse lemmas needed for runtime analysis that relate the grading Γ of a Voronoi diagram to the quality τ. Similar lemmas appear in [HMP06]. They are not novel, but provide a more simple formulation than in prior work.

Lemma 8 (Quality Gives Bounded Grading)**.** *Suppose M is a τ-quality Voronoi diagram. Then for any $x \in \Omega$:*

$$\Gamma_M(x) \lesssim 1$$

Proof. Let $B(c, r)$ be a maximal gap-ball with x on its surface, so that $r = G(x)$. By Lemma 5, $f(x) \gtrsim r = G(x)$. But $\Gamma(x)$ is just the ratio of these values, so it immediately follows that $\Gamma(x) \lesssim 1$. □

Lemma 9 (Bounded Grading Gives Quality)**.** *Suppose we have a Voronoi diagram M, and suppose there is some upper-bound $\gamma \in (0, \infty)$ such that $\Gamma_M(p) \leq \gamma$ at every vertex $m \in M$. Then M is a 2γ-quality Voronoi diagram.*

Proof. Let $p \in M$ be given. Let v be some Voronoi Corner adjacent to $V(p)$. Consider the ball $B(v, R_p)$. This is a gap-ball, thus $G(p) \geq R_p$. Recall from Lemma 3 that $2r_p = f(p)$, so it follows that:

$$\frac{R_p}{r_p} \leq \frac{2G(p)}{f(p)} = 2\Gamma(p) \leq 2\gamma$$

Hence $V(p)$ has aspect-ratio at most 2γ for any $p \in M$, so M is a 2γ-quality mesh. □

Together, these two lemmas allow the discrete notion of Voronoi aspect-ratio and the continuous notion of grading to be freely interchanged in analysis with only a constant factor slack.

2.2.4 Proximity in Well-Spaced Meshes

The following section is concerned with a few simple lemmas about relatively local neighborhoods in a τ-well-spaced mesh. These lemmas essentially state that in a neighborhood consisting of constantly many Voronoi cells, all the cells are relatively the same size, and all the vertices are relatively near one another.

Lemma 10 (Neighbor Sizing)**.** *Suppose M is a τ-well-spaced mesh and vertices u and v are adjacent in M, i.e. $V_M(u) \cap V_M(v) \neq 0$, then:*

- $|uv| \leq 2R_u$
- $R_u \leq \tau R_v$

Proof. The first statement holds for any mesh. Since the cells are adjacent, let $x \in V(u) \cap V(v)$. Then it must be that $|xu| = |xv|$, but then:

$$|uv| \leq |xv| + |xu| = 2|xv| \leq R_v$$

Continuing, since $x \in V(v)$, it must be that $r_u \leq |ux|$, but then by τ-well-spacing:

$$R_u \leq \tau r_u \leq \tau |ux| = \tau |uv| \leq \tau R_v$$

□

Consider then the following corollary:

Corollary 1 (Neighborhood Sizing)**.** *Suppose M is a τ-well-spaced mesh and supposed the shortest path in* $\mathrm{Del}(M)$ *from vertices u to v consists of at most k edges, then:*

- $|uv| \leq 2^k R_u$
- $R_u \leq \tau^k R_v$

Proof. Induction on k using Lemma 10. □

2.3 Scaffolding Preliminaries

This section develops a fairly abstract set of lemmas related to mesh size analysis. Of particular concern is upper-bounding the size of a quality mesh that conforms to set of input points N. Recall that the task of meshing a point set simply requires generating a superset of vertices $M \supset N$ with good-aspect Voronoi cells. This is a vast simplification from conforming to a PLC, but the tradeoff is an abstraction where very strong statements can be made. The lemmas in this section are general-dimensional. Much of this section is joint work with my co-authors, and appears in [MPS08, HMPS09]. This section establishes the theoretical kernel of the Scaffold-Sizing Theorem (Section 4.2). The first lemma is a simple observation:

Lemma 11 (Quality Mesh Upper Bound)**.** *Given a τ-well-spaced point set $M \subset \Omega$:*

$$|M| \lesssim \int_{\Omega} \frac{1}{f_M^d}$$

With constant depending on τ and d.

Proof. A birds-eye of the proof is that $r_m \sim R_m \sim f_M(m)$ at every vertex $m \in M$. So the Voronoi cells roughly pack Ω with cells of volume f_M^d, so the vertices are distributed with *density* $1/f_M^d$. The total *mass* ($|M|$) is the integral of this density.

Formally:

$$\int_{\Omega} \frac{1}{f_M^d} = \sum_{m \in M} \int_{V_M(m)} \frac{1}{f_M^d} \tag{2.3.1}$$

by Lemma 3, $f_M(x) \leq 3R_M$ for all $x \in V_M(m)$, so:

$$\geq \sum_{m \in M} \int_{V_M(m)} \frac{1}{(3R_m)^d} = \sum_{m \in M} \frac{1}{(3R_m)^d} \mathit{Vol}_d(V_M(m)) \tag{2.3.2}$$

but then by definition:

$$\geq \sum_{m \in M} \frac{1}{(3R_m)^d} Vol_d(B_d(m, r_m)) \tag{2.3.3}$$

$$\sim \sum_{m \in M} \frac{1}{(3R_m)^d} (r_m)^d \tag{2.3.4}$$

by τ-well-spaced:

$$\sim \sum_{m \in M} \frac{1}{(3R_m)^d} (R_m/\tau)^d = \sum_{m \in M} \frac{1}{(3\tau)^d} \tag{2.3.5}$$

$$= |M| \frac{1}{(3\tau)^d} \sim |M| \tag{2.3.6}$$

□

Define a few new notions on quality of point sets. The previous notion of quality (namely good aspect-ratio Voronoi) is concerned with a static set of points. The following definitions are dynamic, in that they are concerned with an iteratively growing set of points. The first relates a new point to an existing set:

Definition 23 (θ-medial)**.** *For $\theta \in [0, 1]$, a point x is* **θ-medial** *with respect to a point set M if the distances to the first and second nearest neighbors are within a factor of θ, i.e. $n_M(x) \geq \theta f_M(x)$. x is called exactly-θ-medial when these quantities are equal (see Figure 2.3).*

For a point set M, the vertices $m \in M$ are exactly 0-medial. The Voronoi corners as well as points on the Voronoi edges and facets are exactly 1-medial. This definition is useful when considering a sequence of repeated additions to a point set. Consider that the function f is monotone decreasing as vertices are added to a set, it is natural to ask how well behaved are the changes in f. For some insertion sequences, f can be forced to decrease smoothly, step-by-step.

Definition 24 (Well-Paced Extension)**.** *Given a point set N and an ordered point set $e_i \in E$ with $E \cap N = \emptyset$, define a sequence of supersets: $N_0 := N$, $N_{i+1} := N_i \cup \{e_i\}$. E is a* **θ-Well-Paced Extension** *of N if e_i is θ-medial with respect to N_i for every i.*

This gives rise to the following:

Lemma 12 (Well-Pacing Controls f)**.** *If E is a θ well-paced extension of N, then for all $x \in \Omega$:*

$$f_{N_i}(x) \geq f_{N_{i+1}}(x) \gtrsim f_{N_i}(x)$$

Proof. Let any i be given and take e_i as the vertex added at that step. The left inequality is trivial since $N_i \subset N_{i+1}$. For the bound on the right, consider that if $f_{N_i}(x) \neq f_{N_{i+1}}(x)$, then either $n_{N_{i+1}}(x) = |xe_i|$ or $f_{N_{i+1}}(x) = |xe_i|$. In either case $|xe| \leq f_{N_{i+1}}(x)$. Combine this with two applications of Lipschitz and the θ-medial hypothesis on e_i to obtain:

$$\begin{aligned} f_{N_i}(x) &\leq f_{N_i}(e_i) + |xe| \leq \frac{1}{\theta} n_{N_i}(e_i) + |xe| \\ &= \frac{1}{\theta} f_{N_{i+1}}(e_i) + |xe| \leq \frac{1}{\theta} f_{N_{i+1}}(x) + (1 + \frac{1}{\theta})|xe| \\ &\leq (2 + \frac{1}{\theta}) f_{N_{i+1}}(x) \lesssim f_{N_{i+1}}(x) \end{aligned}$$

□

Statements about well-paced extensions can be made without knowing the base set and the ordered extension explicitly. Consider the following definition:

Definition 25 (Well-Paced Admission)**.** *Given a fixed domain Ω, say that point set N* **admits a (τ, θ)-well-paced ordering** *if there exists some partition of N into (N_0, E), with an E a θ-well-paced extension of N_0. Add the additional constraint that N_0 is τ-well-spaced in Ω. (τ and θ may be suppressed.)*

This sets up an abstract but very strong result in mesh size analysis, the Well-Pacing Theorem:

Theorem 3 (Well-Pacing)**.** *If a pointset N in domain $\Omega \subset \mathbb{R}^d$ admits a well-paced ordering, then:*

$$\int_\Omega \frac{1}{f_N^d} \lesssim |N|$$

Proof. I sketch a proof here, a lengthier proof appears in [MPS08]. The proof proceeds by induction on the well-paced extension. The decrease in f at successive iterations is controlled by Lemma 12. This is shown to control the increase in $\int 1/f^3$ to bounded above a constant between successive iterations. So the final value $\int 1/f_N^3$ is bounded above by $|E| + \int 1/f_{N_0}^3$. Since N_0 was well-spaced by hypothesis, then by Lemma 11, $|N_0| \lesssim \int 1/f_{N_0}^3$. Thus $\int 1/f_N^3 \lesssim |N|$. □

2.4 Surface Reconstruction

This section diverges somewhat from the previous sections of this chapter, which were principally laying groundwork for runtime and sizing analysis. `SVRC` protects creases by laying out a set of spheres to approximate the collar region. `SVRC` then approximates the surface of these spheres by some subset of Delaunay simplices at any time. I take inspiration from the literature on the problem of surface reconstruction: constructing a set of simplices out of a given point set to best represent a surface.

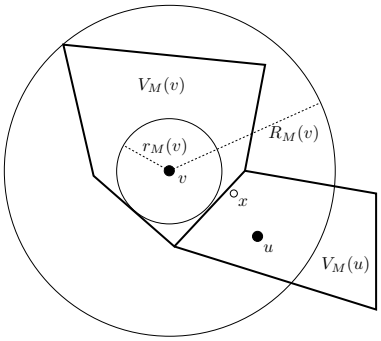

Figure 2.3: The Voronoi cells of two vertices u and v in a vertex-set M (not pictured). The radii of the inner-ball and outer-ball of v are labeled. The point x is 0.9-medial.

2.4.1 Collar Systems

The region that `SVRC` constructs is a **collar system**.

Definition 26 (Collar System)**.** *A* **collar system** $\mathcal{B}$ *is a set of* **collar balls** $B(c,r) \in \mathcal{B}$*. Define the* **collar surface** $\mathcal{S}$ *as the boundary of* $\bigcup \mathcal{B}$*. $\mathcal{S}$ must have the following requirement: If a point $s \in \mathcal{S}$ intersects more than one ball, then s must be contained in some circle C with $C \subset \mathcal{S}$. This disjointness property guarantees that $\mathcal{S}$ can be described as a collection of circles $\mathcal{C}$ and partial spheres $\mathcal{A}$, call this pair $\langle \mathcal{C}, \mathcal{A} \rangle$, the* **collar description** *of a collar surface.*

A collar system is also be defined in two dimensions, in this case the collar description is as arcs and points. Consider taking a subsystem of a three-dimensional collar system by taking only those collar balls whose centers lie on a common plane. The intersection of the collar subsystem and the plane yields a new collar system in two dimensions. A collar system in one dimension is just a set of points.

`SVRC` will have a set of balls that is a subset of the collar region. It is always a collar system by the following:

Lemma 13 (g-sized balls form a system)**.** *Suppose $\mathcal{P}$ is a PLC and $\mathcal{B}$ is a set of balls centered on the creases of $\mathcal{P}$ satisfying $r < g(c)/2$ for every $B(c,r) \in \mathcal{B}$. Then $\mathcal{B}$ is a collar system.*

Proof. Balls are centered on creases, so every ball is either centered on a vertex or a segment. First claim that any two balls in $\mathcal{B}$ only intersect if they are centered on the same segment. If u and v are the centers of two balls not centered on the same segment, then the crease containing u does not contain the crease containing v and vice-versa, so $g(u) \leq |uv|$ and $g(v) \leq |uv|$. But by assumption $r_v + r_v < (g(u) + g(v))/2 \leq |uv|$ so they cannot intersect.

So if a point s on the surface $\mathcal{S}$ of $\mathcal{B}$ is in more than one ball, then all these balls intersecting s, must be centered on the same segment. Consider the plane normal to the segment through s, its intersection with $\mathcal{S}$ is an appropriate circle. □

It will also be useful to have some geometric control over the collar system. To this end, define a **Smooth Collar System** as follows:

Definition 27 (Smooth Collar System)**.** *A* **smooth collar system** *is a collar system such that the any two balls whose surfaces intersect meet with at a tangent angle at least $\pi/2$.*

Such collar systems can be characterized simply:

Lemma 14. *A collar system is a smooth collar system iff for every pair of balls $B(c_1, r_1)$ and $B(c_2, r_2)$ with intersecting surfaces, it holds that:*

$$|c_1c_2|^2 \leq r_1^2 + r_2^2$$

Proof. Follows from straightforward trigonometry, in particular the law of cosines. Consider two intersecting balls, and let a point x on both their surfaces. Consider the triangle formed by x, c_1, and c_2. The edges (x, c_1) and (x, c_2) are normal to their respective spheres, so the angle between these edges is the supplement of the tangent angle at x. So the lemma is satisfied if the angle α between these edges is at most $\pi/2$. The law of cosines states:

$$|c_1c_2|^2 = |xc_1|^2 + |xc_2|^2 - 2|xc_1||xc_2|\cos\alpha = r_1^2 + r_2^2 - 2r_1r_2\cos\alpha$$

α is at most $\pi/2$ precisely when $\cos\alpha \geq 0$, which occurs iff the hypothesis is true. □

2.4.2 Restricted Delaunay and Voronoi

A hallmark definition for surface reconstruction theory is the restricted Delaunay diagram of a set of vertices restricted to a surface. For simplicity, I will only consider collar surfaces, but the definition is general:

Definition 28 (Restricted Delaunay and Voronoi of a Collar Surface)**.** *Let a point set M and a collar surface $\mathcal{S}$ be given. Define the* **Restricted Voronoi diagram** *of M restricted to $\mathcal{S}$, denoted $\mathrm{Vor}(M)|_{\mathcal{S}} \subset \mathrm{Vor}(M)$, as the subset of Voronoi polytopes that intersect $\mathcal{S}$:*

$$\mathrm{Vor}(M)|_{\mathcal{S}} := \{V \in \mathrm{Vor}(M) | V \cap \mathcal{S} \neq \emptyset\}$$

Define the **Restricted Delaunay Diagram** *($\mathrm{Del}(M)|_{\mathcal{S}} \subset \mathrm{Del}(M)$) as the dual simplices:*

$$\mathrm{Del}(M)|_{\mathcal{S}} := \{\mathrm{Dual}(V) \mid V \in \mathrm{Vor}(M)|_{\mathcal{S}}\}$$

See Figure 2.4.2.

The restricted Delaunay definition also applies for collar surfaces in lower dimension. The restricted Delaunay definition implicitly assumes the following non-degeneracy condition on the collar description and the point-set: no Voronoi corner lies on a partial sphere; no Voronoi

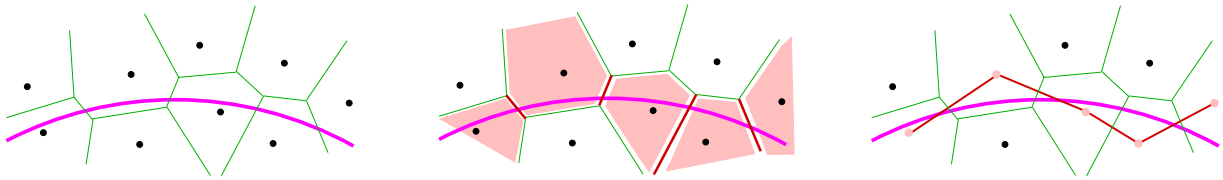

Figure 2.4: *Left:* A Voronoi diagram $\mathrm{Vor}(M)$ and a surface $\mathcal{S}$ cutting through. *Center:* The restricted Voronoi cells $\mathrm{Vor}(M)|_{\mathcal{S}}$. *Right:* The restricted Delaunay triangulation $\mathrm{Del}(M)|_{\mathcal{S}}$ approximates $\mathcal{S}$.

facets or edges are tangent to a sphere; no Voronoi corners or edges intersect a circle; and no Voronoi facets are tangent to a circle. The exact radii of a collar balls is never critical in `SVRC`, so this can always be achieved by perturbation.

Observe that if any vertices of M are actually on $\mathcal{S}$, they will always be contained in the restricted Delaunay. The restricted Delaunay is closed under subsets: if $\mathcal{S}' \subset \mathcal{S}$, then $\mathrm{Del}(M)|_{\mathcal{S}} \subset \mathrm{Del}(M)|_{\mathcal{S}'}$. So it makes sense to associate portions of the restricted Delaunay with pieces of the collar description. (Note that this does not yet necesarily partition the restricted Delaunay, some simplices may be associated with more than one sphere or circle.)

I will employ the following definitions regarding $\mathrm{Del}(M)|_{\mathcal{S}}$:

Definition 29 (Representation Sets)**.** *Consider M, $\mathcal{S}$ and $\mathrm{Del}(M)|_{\mathcal{S}}$ as above. For a simplex $T \in \mathrm{Del}(M)|_{\mathcal{S}}$, define its* **representation set** $\mathrm{Rep}(T) := \mathrm{Dual}(T) \cap \mathcal{S}$*. For a triangle, the dual is a voronoi edge, so this yields a set of* **representation points***. For an edge, call it a* **representation curve***. For a vertex, call it a* **representation patch***. (Note that these sets may not actually be a single curve or patch.)*

Representation sets are also defined in two dimensions. The restricted Delaunay does a good job approximating a surface. Consider, that as more and more points are drawn from $\mathcal{S}$ and added to M, $\mathrm{Del}(M)|_{\mathcal{S}}$ converges to $\mathcal{S}$ in several norms. The main goal is to show is that $\mathrm{Del}(M)|_{\mathcal{S}}$ has the same topology as the collar surface. The chief technology employed in such results is based on the theorem of Edelsbrunner-Shah[ES97], which requires the so-called topological-ball property, that the intersection of each representation set with the surface is a topological ball of proper dimension. `SVRC` will achieve this by enforcing a set of topological constraints.

Definition 30 (Consistent Sample)**.** *Given a point set M and a collar surface $\mathcal{S}$, a simplex $T \in \mathrm{Del}(M)|_{\mathcal{S}}$ is called* **locally consistent** *if $\mathrm{Rep}(T) \cap \mathcal{S}$ is a topological $(|T|-1)$-ball. M is a* **consistant sample** *for $\mathcal{S}$ if every simplex of $\mathrm{Del}(M)|_{\mathcal{S}}$ is locally consistent.*

Concretely, this means that every set of representation points must be a singleton, every representation curve must be topological 1-ball, and every representation patch must be topological 2-ball.

A restricted Delaunay simplex T can always verify its local-consistency by testing on only the subset of collar balls that intersect Dual(T). Note that if M is a consistent sample of $\mathcal{S}$, this does not necessarily imply that M is a consistent sample of some subset of $\mathcal{S}$.

Definition 31 (Collar-Consistent Sample)**.** *A consistent sample M of $\mathcal{S}$ is a* **collar-consistent sample** *if M is a consistent sample for the set of circles in the collar decomposition of $\mathcal{S}$.*

Lemma 15 (Collar Surface Topology)**.** *If M is a collar-consistent sample for a collar surface $\mathcal{S}$, then $\text{Del}(M)|_{\mathcal{S}}$ is a triangulation homeomorphic to $\mathcal{S}$. Furthermore, M agrees with the collar decomposition, so that the restricted Delaunay of M restricted to any partial sphere is a triangulation homeomorphic to the partial sphere, and the restricted Delaunay of M restricted to any circle forms a cycle.*

Proof. Since M is a consistent sample, it will meet the topological ball property of the Edelsbrunner-Shah [ES97], which will guarantee that $\text{Del}(M)|_{\mathcal{S}}$ is homeomorphic to the collar surface. Since M is a collar-consistent sample, then for the set of circles $\mathcal{C}$ in the collar description, $\text{Del}(M)|_{\mathcal{C}})$ is a set of cycles. By nesting of the restricted Delaunay, this is a subset of $\text{Del}(M)|_{\mathcal{S}}$. These cycles partition $\text{Del}(M)|_{\mathcal{S}}$ in to the restricted Delaunay of each partial sphere. □

For termination purposes, `SVRC` must eventually obtain a collar consistent sample. This is will be guaranteed using the notion of ε-sampling. For general surfaces, this requires the notion of a medial-axis and a local feature size [Dey06]. For collar surfaces, I develop simpler definitions:

Definition 32 (Collar Surface Sizing)**.** *Consider a collar system $\mathcal{B}$ with $\mathcal{S}$. For $s \in \mathcal{S}$:*

$$rad(s) := \min\{r \mid B(c,r) \in calB \text{ and } s \in B\}$$

Define $css : (\mathcal{S} \to \mathbb{R})$ as the largest Lipschitz function that is less than or equal to $rad(s)$

Viewing any single piece (circle or partial sphere) of the collar description, this definition guarantees that *css* at any point on the piece will be less than the distance to the medial axis of the piece.

Definition 33 (ε-Sample of a collar surface)**.** *A point set $M \subset \mathcal{S}$ is an ε-sample of $\mathcal{S}$ if the following:*

- $\forall\ s \in \mathcal{S},\ |sM| \leq \varepsilon css(s)$
- *For every circle C in the collar description of $\mathcal{S}$, setting $M_C := M \cap C$, it holds that:*

$$\forall\ s \in C,\ |sM_C| \leq \varepsilon css(s)$$

The second condition is motivated by the non-smoothness of the collar surface around the circles, requiring that there be sample points M_C on the circles.

The following lemma will then be employed later for termination:

Lemma 16 (ε-Sample gives Consistent Sample)**.** *There exists $\varepsilon \in (0, 1)$ such that if M is an ε-sample of a smooth collar surface $\mathcal{S}$, then M is a collar-consistent sample of $\mathcal{S}$.*

Proof. When M is an ε-sample, $\mathcal{S}$ must be nearly flat in a relative neighborhood of any vertex v of M, then the intersection of $\mathcal{S}$ with any Voronoi cell will simply be the intersection of a plane (or line) with a convex polytope, so it will have the proper topology. For details, see [Dey06], Chapter 3. □

The algorithm `SVRC` also seeks to approximate the geometry of the collar surface, motivating the following:

Definition 34 (Representation Angle)**.** *Consider M, $\mathcal{S}$, and a triangle $T \in \mathrm{Del}(M)|_{\mathcal{S}}$. $\mathrm{Dual}(T)$ is a Voronoi edge E that pierces $\mathcal{S}$ at $\mathrm{Rep}(T)$. For a point $p \in \mathrm{Rep}(T)$, p is on some partial sphere S of the collar description. The* **representation angle** *$\sigma(T, M, \mathcal{S})$ is measured at p as the angle between E and the normal of S at p. If $\mathrm{Rep}(T)$ is not a single point, take the largest (worst) representation angle.*

The representation angle gives a measurement of how tilted a triangle T is relative to the local portion of the collar surface $\mathcal{S}$. Note that by perpendicularity, σ is also the angle between the plane containing T and the plane tangent to $\mathcal{S}$ at $\mathrm{Rep}(T)$ Furthermore, if all the vertices of T lie on the same partial sphere S, then $\sigma = 0$.

Representation angles are defined analogously for edges in two dimensions The notion of representation angle is not defined in general for edges in three dimensions. However, in the specic case when C is a circle of the collar description and $E \in \mathrm{Del}(M)|_C$, then I define the representation angle $\sigma(E, M, C)$ as follows: $\mathrm{Dual}(E)$ is a facet that intersects C at some point $p = \mathrm{Rep}(E)$; measure the angle between the normal of C at x and the plane of $\mathrm{Dual}(E)$. This angle is measures how well E approximates the tangent of C at $\mathrm{Rep}(e)$.

Algorithm `SVRC` will add points on the collar surfaces to guarantee collar consistency and to obtain a bound on representation angles. The bound on representation angles is used to guarantee the quality of simplices that are created adjacent to creases.

Chapter 3

Algorithm SVRC

The algorithm SVRC is based primarly on the sparse refinement paradigm developed in SVR [HMP06]. The input to SVRC is a PLC $\mathcal{P}$, and two constants. The output is a tetrahedralization $\mathcal{T}$ that conforms to $\mathcal{P}$. $\mathcal{T}$ has bounded radius-edge ratio tetrahedra everywhere except adjacent to creases. Near the creases, $\mathcal{T}$ contains tetrahedra with no large dihedral angles. The runtime of SVRC is near-optimal $O(|\mathcal{P}| \log \Delta + |\mathcal{T}|)$. The total space usage is optimal $O(|\mathcal{T}|)$.

The constants input to SVRC are τ and σ. τ governs the radius-edge ratio of the tetrahedra away from creases, and must be given with $\tau > 4\sqrt{2}$, in order to later satisfy Theorem 5. σ gives a guarantee on the largest dihedral angle for output tetrahedra adjacent to a crease. Two internal parameters are constructed, θ and θ_0. The constant θ must generally be chosen very close to 1 in order to later satisfy Theorem 5, and the constant θ_0 is always prescribed as $\theta_0 := 2\theta/(1 + 2\theta)$, which is then about 2/3. θ can be set internally closer to 1 to control a tradeoff between runtime and output-size.

This chaper presents a high-level overview of SVRC, followed by a description of implementation details, and then detailed pseudocode for the algorithm.

3.1 Algorithm Overview

SVRC is an iterative algorithm. A very bland mesh is created to begin with, and vertices are added to this over time until a correct (i.e. quality, conforming) mesh is achieved. In actuality, SVRCcreates several very bland meshes, one for each input feature. For $F \in \mathcal{P}$, these will each be denoted M_F. One three-dimensional volume mesh M_Ω is also created. The goal of SVRC is to refine M_Ω until it is suitable for output.

Before the meshes can be created, SVRC creates $dim(F)$-dimensional domains Ω_F. For input vertices and segments, $\Omega_F = F$. For an input facet F, a **bounding-box** Ω_F is chosen from $\mathbb{R}^2$ meeting the following criteria:

Definition 35 (Bounding Box)**.** *Consider $X \subset \mathbb{R}^d$ for $d \geq 2$, let L be the diameter of X, $\Omega_X \subset \mathbb{R}^d$ is a* **bounding-box** *if:*

- *Ω_X is bounded, convex, and $X \subset \Omega_X$*
- *The diameter of $\Omega_X \lesssim L$*
- *The distance from any point in X to the exterior $(\mathbb{R}^d - \Omega_X)$ is $\gtrsim L$*

The second condition is to ensure that a bounding-box is not too large, since a triangulation or tetrahedralization will have to cover the whole thing. The third condition is to prevent any interactions with the border of a bounding-box. `SVRC` takes Ω_F as a square centered on F of an appropriate diameter for each 2D feature F. Take $\Omega = \Omega_{\mathcal{P}}$ as a cube centered on $\mathcal{P}$ of the an appropriate diameter.

3.1.1 Sparse Refinement

This subsection describes how `SVRC` approaches the basic sparse refinement paradigm. The next subsection (3.1.2) explains the significant changes in `SVRC` to handle creases.

Each mesh M_F is initialized with a constant number of points at the boundary of Ω_F. `SVRC` then proceeds by queueing up **work events**, gap-balls needing destruction in a given mesh. A work event destorys a gap ball by adding a new vertex to the mesh inside the gap ball, refining the meshes until all the work is done Work events are queued for three main reasons: SKINNY, UNRESOLVED, and ENCROACHED.

The work event type SKINNY is simple. At termination, `SVRC` wants every mesh to be τ-well-spaced. If there is some skinny Voronoi cell, then it is adjacent to a gap-ball that is much too large. The algorithm will destroy this gap-ball by inserting a vertex relatively near its center.

The second type is for purposes of conforming to $\mathcal{P}$. Each feature mesh makes sure it conforms to its own feature, and then globally conforming subdivision of $\mathcal{P}$ is decided upon by `SVRC` at the lowest dimensions and recursively enforced upward. This motivates the notion of **containment dimension**.

Definition 36 (Containment Dimension and Parent Feature)**.** *The* **containment dimension** *of a point $x \in \Omega$ is the dimension of the lowest-dimensional feature domain containing x:*

$$\mathrm{CD}(x) := \min\{dim(F) \mid F \in \mathcal{P} \text{ and } x \in \Omega_F\}$$

For a simplex T, containment dimension is given by the highest among points in T, so $\mathrm{CD}(T) := \max_{x \in X} \mathrm{CD}(x)$ A ball inherits the containment dimension of its center, so $\mathrm{CD}(B(c, \_)) := \mathrm{CD}(c)$. The **parent feature** *of x is the feature $F \in \mathcal{P}$ with $dim(F) = \mathrm{CD}(x)$ and $x \in \Omega_F$.*

If there is ever a disagreement between feature meshes, then there is a feature mesh that does not contain some simplex of a subfeature mesh. (Consider that initially, M_Ω does not contain any of the input vertices, which are 0-simplices of their own feature meshes.) Such a simplex is **unresolved** in the superfeature mesh. `SVRC` finds large gap-balls in a mesh containing unresolved simplices and queues these as work events.

A gap-ball B is always destroyed by inserting some p near the center of B. The first choice to destroy B is to add its center c, but if there is an unresolved vertex v nearby c with $\mathrm{CD}(v) < \mathrm{CD}(c)$, this would be even better, so `SVRC` will **warp** the insertion of c to v. If all the unresolved simplices are too far from c, then `SVRC` cannot warp. It will destroy the gap-ball by inserting c, then find new, smaller gap-balls containing the same still-unresolved simplices and queue new work events.

The parameter $\theta \in (0, 1)$ controls how far `SVRC` is allowed to warp. If $\theta = 0$, the algorithm will never warp, never conform to $\mathcal{P}$ and never terminate. If $\theta = 1$, then `SVRC` will always warp to the unresolved simplex, but guarantees on asymptotic runtime disappear. Always warping will tend to favor adding fewer total vertices. Thus θ becomes a tradeoff between runtime and output-size. This phenomenon is observed empirically in [AHMP07].

The third type of work event ENCROACHED is related to both sizing and conforming. `SVRC` should not return a mesh with arbitrarily many points. If a point is arbitrarily added ε away from to a feature F, then a mesh may not be able to be conform without inserting $\log(1/\varepsilon)$ points. `SVRC` desires that the final number of points be related somehow to $f_{\mathcal{P}}$, so this must be avoided. To do so, `SVRC` puts a set of protective balls with low containment dimension around each subfeature. A protective ball B is **encroached** by a point p if p intersects the interior B°. If `SVRC` is trying to destroy some B by inserting p and it happens that p encroaches on a protective ball B' with $\mathrm{CD}(B') < \mathrm{CD}(p)$, the insertion is disallowed. B still must be eventually destroyed, however, so `SVRC` **yields** and destroys B' first, before re-attempting to destroy to B.

The distinction between warping and yielding is a bit subtle. The goal is to keep the meshes in sync with one another with regards to conforming. Yielding prevents a high-dimensional mesh from getting too far ahead with regards to refinement, it must slow down by refining lower-dimensional meshes. Warping prevents a high-dimensional mesh from getting too far behind, it should catch up by inserting unresolved points from lower-dimensional meshes. The main procedure for destroying gap-balls, warping, and yielding is found in DESTROY (Figure 3.3).

3.1.2 Enhancements for handling Creases

The first thing to note is that when $\mathcal{P}$ has sharp creases, a point set cannot ever be conforming and well spaced, so `SVRC` so far would run forever. The solution taken by `SVRC` is to give up on conforming near creases for the run of the algorithm, and then stitch in a conforming mesh around the creases as a post-process. The area around the creases where `SVRC` aban-

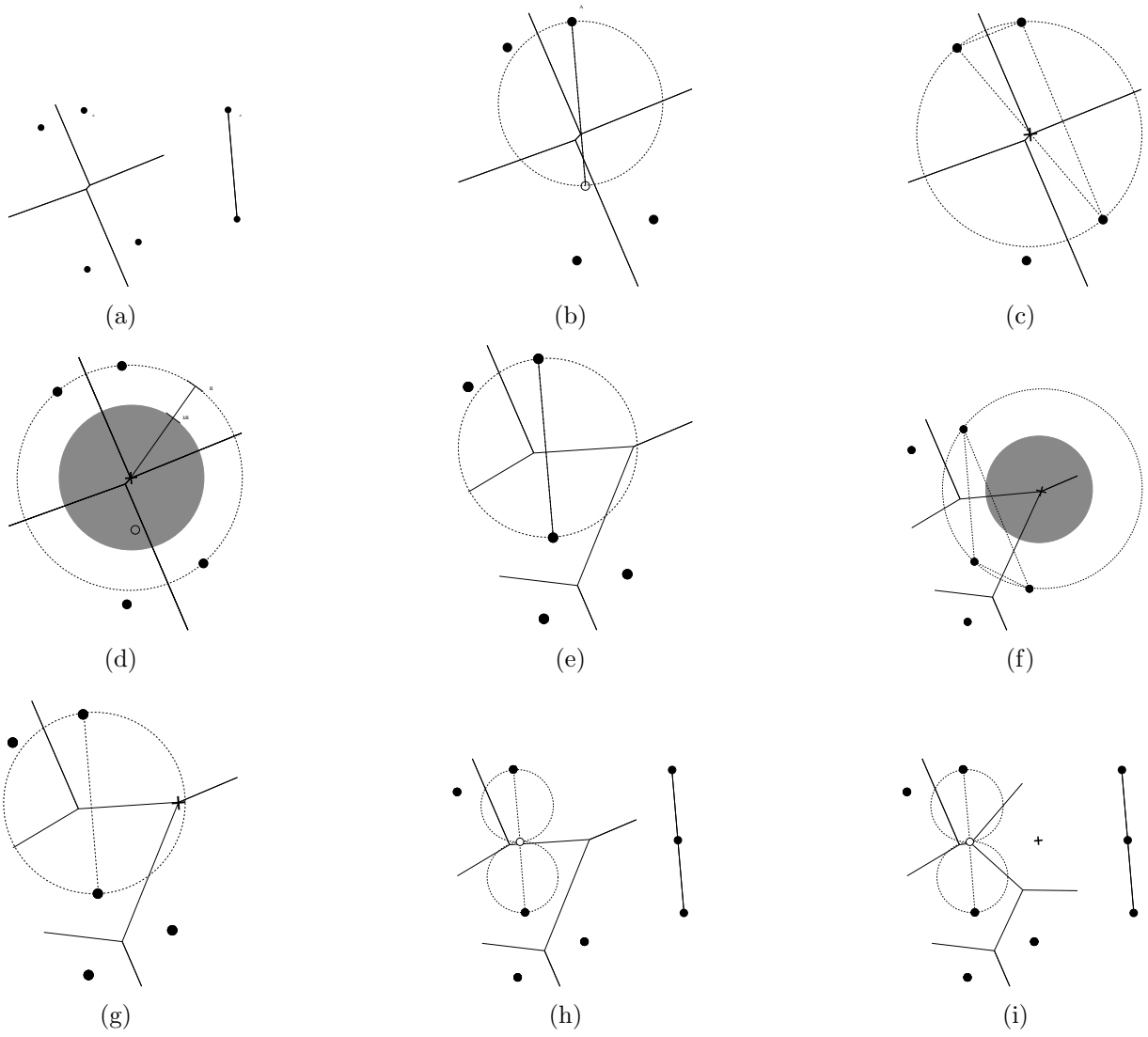

Figure 3.1: The sparse refinement process. (a) A 2D mesh and a 1D mesh side by side. The vertex v is resolved in both meshes. (b) The intersections between 1D protective balls and the 2D mesh are stored in the Bipartite Location Graph (BLG). (c) A work event for a skinny element desires the insertion of a circumcenter/Voronoi corner. (d) `SVRC` looks for any unresolved vertex nearby (within θ times the radius) in order to warp if necessary. (e) It finds the unresolved 1D vertex and inserts this instead, updating the Voronoi diagram and the BLG. (f) A new work event for a skinny cell finds no vertices to warp to. (g) However, the new insertion would encroach on a 1D protective ball, so yield by refining the 1D mesh instead. (h) The 1D mesh is refined, and the BLG is updated for new intersections. (i) The 2D mesh resumes its insertion, which now finds to vertices to warp to and no encroached protective balls.

dons conforming is a collar system. The algorithm simply discards any UNRESOLVED work events if they are inside the collar system. This is enough to guarantee termination, quality everywhere, and conforming everywhere outside the collar system.

Unfortunately, a collar system is not known in advance, so the algorithm must create a collar system and dynamically add to it. Any UNRESOLVED work events in the *current* collar system are discarded, except to recover the interior of a segment crease. As long as a whole collar system approximating the collar region is recovered early enough, `SVRC` can get the same guarantees as if it knew the collar region in advance.

Portions of the collar system are recovered whenever M_Ω adds a vertex v that lies on a crease. At insertion time, v calculates its sizing $g(v)$ via the routine FASTCOLLARSIZE (Figure 3.3 Then is adds a new collar ball $B(v, g(v)/3)$ to the collar system. If v was inserted in time to ensure that $n_{M_\Omega}(v) \gtrsim g(v)$, then `SVRC` has not refined too far past the collar surface. Overall, the mesh size in an area will only be going down when `SVRC` deals with UNRESOLVED events. If care was not taken, `SVRC` might continually warp to points on the features adjacent to the crease, rather than warping to the crease itself (which would heroically augment the collar system and prevent infinite refinement). To give the crease precedence, `SVRC` always warps to the lowest-dimensional unresolved vertex.

Once all refinement has stopped, `SVRC` should have an collar-consistent sample of the collar surface (recall Section 2.4). This is ensured by enqueuing work events of a fourth type: COLLAR. The restricted Delaunay of the collar surface is maintained, and if it is ever not locally consistent, `SVRC` enqueues a work event to add a vertex on the collar surface by destroying some gap-ball centered on the surface. It is important to conform to the collar system, but less important than conforming to features. To reflect this, redefine the containment dimension for points on the collar surface as follows:

Definition 37 (CD for collars)**.** *Consider the collar description of circles and partial spheres. Let the "collar dimension"* $d(x)$ *of a point* x *be* 1 *if* x *is on a circle,* 2 *if* x *is on a partial sphere, or* 3 *if* x *does not intersect the collar surface. Then redefine the containment dimension of a point* x *by the pair* $\mathrm{CD}(x) := \langle \mathrm{CD}(x), d(x) \rangle$ *Ordering on containment dimension is then given by lexicographical ordering on the pairs, i.e. so* $\mathrm{CD}(x) \leq \mathrm{CD}(y)$ *if the old containment dimension was less, or if the old containment dimension was the same and* x *has lower collar dimension.*

Recall the collar subsystem intersecting an input facet is described by arcs and points, and the intersection with an input segment is just a points. This sets up the following ordering by CD from lowest to highest:

(0,3) input vertices
(1,2) points where collar spheres intersect input segments
(1,3) points on input segments
(2,1) points where collar circles intersect input facets

(2,2) points where collar spheres intersect input facets

(2,3) points on input facets

(3,1) points on collar circles

(3,2) points on collar spheres

(3,3) points in space

Note that a few cases do not exist, since collar surfaces never intersect input vertices, and collar circles never intersect input segments. This priority is used for enforcing conforming and good collar approximation across all the meshes. Insertions always warp or yield to lower containment dimension balls.

An additional procedure is added to the algorithm to handle vertices that may be very near the surface of a collar ball. A constant factor $0 < \delta << 1$ is selected. The factor δ must be smaller than $\varepsilon \sin(\pi/2 - \sigma/2)$, where ε is the constant in the proof of Lemma 16. For a collar ball with radius R, whenever a vertex is within δR of the surface, the vertex is symbolically snapped to SNAP performs a symbolic perturbation of the coordinates of v. For all Delaunay and Voronoi calculations, v retains its original calculations. For calculating the quality of the collar approximation, the warped coordinates are used. This ensures that SVRC will not attempt to fill the tiny gap between the vertex and the sphere only for the sake of collar approximation. The choice of δ ensures that the perturbation does not significantly disturb the representation angle in the restricted Delaunay.

Once all work events are complete, SVRC deletes all the vertices between the creases and the implicitly represented collar surface. A new tetrahedralization is then manually inserted by SVRC to fill the collar system. This procedure is shown in Figure 3.3. The new tetrahedra will have no-large-angles so long as the implicit surface has good representation angles. To ensure good representation angles at the finale, work events are also queued with reason COLLAR whenever the restricted Delaunay contains a representation angle worse than the parameter $\sigma \in (0, \pi/2)$. Setting σ near zero will yield dihedral angles not much larger than $\pi/2$, which will require a great deal of vertices on the collar surface, so σ becomes a tradeoff between quality of simplices adjacent to collars and overall mesh size.

3.1.3 Correctness of SVRC

Assuming that SVRC terminates, then there no more work events of any of the four types UNRESOLVED, ENCROACHED, SKINNY, and COLLAR. The following claim of correctness can then be made:

Theorem 4 (SVRC Quality and Conforming)**.** *Given a PLC P, if SVRC terminates, it outputs a conforming tetrahedralization $\mathcal{T}$ with good radius-edge tetrahedra everywhere except adjacent to creases. Tetrahedra adjacent to creases have no-large-angles.*

Proof. The first claim is that SVRC generates a tetrahedralization. Let $\mathcal{B}$ be the final collar system with collar surface $\mathcal{S}$. Let $M := M_\Omega$ immediately before the call to post-process RipStitchCollar. M is a tetrahedralization of Ω. Let $\mathcal{T}' = \text{Del}(M)|_\mathcal{S}$, note $\mathcal{T}' \subset \mathcal{T}$. Since there are no more Collar events, M is a collar-consistent sample for $\mathcal{S}$, so by Lemma 15, $\mathcal{T}'$ is a triangulation homeomorphic to $\mathcal{S}$. $\mathcal{S}$ divides Ω into two closed regions, "inside" ($\mathcal{B}$) and "outside" ($\Omega - \mathcal{B}$) whose intersection is $\mathcal{S}$. By homeomorphy, $\mathcal{T}'$ does the same, call these closed regions $\mathcal{I}$ and $\mathcal{O}$. The inside $\mathcal{I}$ is tetrahedralized by RipStitchCollar, the outside $\mathcal{O}$ is tetrahedralized by $\text{Del}(M \cap \mathcal{O})$, and the intersection is the triangulation $\mathcal{T}'$, so $\mathcal{T}$ is a tetrahedralization of Ω.

Since there are no more work events of type Skinny or Collar, then the quality guarantees trivially hold. The primary issue is conforming. Since there are no Unresolved events interesecting $\mathcal{O}$, $\text{Del}(M)$ is outside-conforming in the sense that for any feature $F \in \mathcal{P}$, $F \cap \mathcal{O}$ appears as a union of simplices of $\text{Del}(M)$, Accordingly, $F \cap \mathcal{T}'$ is a union of simplices of $\mathcal{T}'$. It only remains to conform inside $\mathcal{I}$. Let $F \in \mathcal{P}$ that intersects $\mathcal{I}$, then it remains to show that $F \cap \mathcal{I}$ must be a union of simplices from $\mathcal{T}$. Suppose F is a crease, then F is resolved by RipStitchCollar. so $\mathcal{T}$ conforms to F.

Suppose F is not a crease. If F is a segment, it enters $\mathcal{I}$ at some point v in $\mathcal{T}'$ and is adjacent to a crease vertex v'. This edge was added so $\mathcal{T}$ conforms to F If F is a facet, then it enters $\mathcal{I}$ along some path in $\mathcal{T}'$ and is adjacent to some crease segments. A set of triangles stitching this crease segment to the path is added by RipStitchCollar, so $\mathcal{T}$ conforms to F.

Thus $\mathcal{T}$ is a conforming tetrahedralization of $\mathcal{P}$. □

3.2 Structures in SVRC

3.2.1 Data Structures

For each mesh M_F, use a cell-complex data structure to store the whole of $\text{Vor}(M_F)$. The cell-complex should be able to handle Voronoi insertions in time proportional to the number of cells that are changed. Vertices are augmented to answer if they lie on a crease of $\mathcal{P}$.

Maintain 11 total work queues, one for each of the four types of work events paired with relevant dimensions. Voronoi cells of any dimension but 0 may be skinny, so maintain $\text{SkinnyQueue}_{1\ldots3}$. Voronoi cells of dimension > 0 will need to resolve subfeature meshes, so maintain $\text{UnresolvedQueue}_{1\ldots3}$. Diametral balls only protect edges and facets, so maintain $\text{EncroachedQueue}_{1\ldots2}$. The collar system is projected to subsystems in each lower dimension, each having their own collar surface, so maintain $\text{CollarQueue}_{1\ldots3}$.

Every work event popped off a queue, causes a call to Destroy which handles warping and yielding before eventually calling ForceInsert to perform an actual insertion into a mesh. If a point is inserted on a collar, a call is spun off to CreateCollar to augment the

collar system appropriately.

Managing the work queues is done with ENQUEUE and DEQUEUE (Figure 3.3). DEQUEUE pops an event in particular priority order on the type and dimension. This ordering is crucial for efficiency (Chapter 5). But for correctness, any order on the work events will suffice. Each work event is a gap-ball represented as BALLS.

3.2.2 Protective Balls

SVRC uses a system of protective balls to represent the work events. For the purposes of SVRC, a BALL is a pair $\langle (c, r), F \rangle$, where (c, r) is a center and radius and $F \in \mathcal{P}$ is a feature for which this is a gap-ball. The BALLS are divided into two groups, REPBALLS and CONFBALLS. REPBALLS are **representation balls** and are used for the implicit collar surface representation (recall Section 2.4):

Definition 38 (Representation Balls)**.** *Consider a 3-dimensional mesh M and collar surface $\mathcal{S}$. Let $\langle \mathcal{C}, \mathcal{A} \rangle$ be the collar description of $\mathcal{S}$. The* **representation balls** *of $\mathrm{Del}(M)|_{\mathcal{S}}$ are given by:*

1. *For every $A \in \mathcal{A}$, for every triangle $T \in \mathrm{Del}(M)|_{\mathcal{A}}$ with a vertex v*

$$\{B(c, |cv|) \mid c \in \mathrm{Rep}(T)\}$$

2. *For every $C \in \mathcal{C}$, for every edge $E \in \mathrm{Del}(M)|_{\mathcal{C}}$ with a vertex v*

$$\{B(c, |cv|) \mid c \in \mathrm{Rep}(E)\}$$

Consider a two-dimensional mesh M' and collar surface $\mathcal{S}$. Let $\mathcal{A}$ and $\mathcal{P}$ be the arcs and points of the collar description of $\mathcal{S}$. Then the representation balls of $\mathrm{Del}(M')_{\mathcal{S}}$ are given by:

1. *For every $A \in \mathcal{A}$, for every edge $E \in \mathrm{Del}(M)|_{\mathcal{A}}$ with a vertex v*

$$\{B(c, |cv|) \mid c \in \mathrm{Rep}(E)\}$$

2. *For every $p \in \mathcal{P}$, $B(p, 0)$*

Consider a one-dimensional mesh M'', its collar surface is just a set of points P. The representation balls are $\{B(p, 0) \mid p \in P\}$.

Since REPBALLS inherit the containment dimension of their centers, and their centers are on the collar surface, they have fractional containment dimension. Balls of type 2 in the definition have lower containment dimension than those of type 1.

From the definitions of restricted Delaunay and representation sets, any representation ball will be an empty ball centered on the collar surface. Note that these balls are guaranteed empty by definition. Whenever the restricted Delaunay is changed, the new RepBalls will be empty, even if they are not nested inside the old RepBalls.

RepBalls manage the implicit representation of the collar surface. If there are bad representation angles or bad topology, then the appropriate RepBall is queued for destruction, which will add its center, a new point on the collar surface.

The other balls are conforming balls (ConfBalls). These protect lower-dimensional features so that higher dimensional meshes will conform to them. To protect a lower-dimensional simplex S, `SVRC` uses its diametral ball. This works since if the vertices of S are in M, and the diametral ball of M is empty, then $S \in \mathrm{Del}(M)$. The ConfBall to protect a vertex v is a 0-radius ball. If a ConfBall is encroached by a higher-dimensional mesh, then `SVRC` will destroy it by adding its center to the mesh of the lower-dimensional feature it is protecting.

For reasons of runtime, `SVRC` might not enqueue a ConfBalls for destruction immediately after it is encroached, preferring to wait until it is **doubly-encroached** by two vertices. For this reason, ConfBalls store an flag PartiallyEncroached which may be set true or false.

3.2.3 Object Location

The main work in `SVRC` is Destroy and ForceInsert. Destroy handles warping and yielding when destroying a gap-ball, eventually calling ForceInsert performs an actual mesh insertion (Figures 3.3 and 3.3). This requires supporting several local operations. In particular, these methods need to find out about any lower-dimensional work events that are in a local neighborhood. This work is principally handled by a bipartite graph called the Bipartite Location Graph (BLG). The BLG is an intersection graph, edges are pairs of geometrically intersecting objects.

On one side of the BLG are location objects, and on the other side are all the Voronoi cells of all the feature meshes. Location objects are of two varieties, Balls and Collars. Balls are the RepBalls and ConfBalls discussed in Section 3.2.2. Collars are collar balls and circles and are used to track intersection of the collar system with the meshes. (In a two dimensional mesh, Collars are circles and intersection points.)

The edges of the BLG are directed pointers, labelled and grouped by the type of objects. Consider first edges from location objects to cells. Suppose B is a Ball. Recall that B is actually a pair containing a ball and a feature F. If $\mathcal{F} \subset \mathcal{P}$ are the superfeatures of F, then for each $F' \in \mathcal{F}$, there is a bundle of pointers given by $\textsc{Cells}_{F'}(B)$ that are pointers to the cells in $\mathrm{Vor}(M_{F'})$ that intersect B. For every feature $F \in \mathcal{P}$, a Collar C has a bundle of pointers $\textsc{Cells}_F(C)$ for the cells in $\mathrm{Vor}(M_F)$ intersecting C.

Now consider pointers in the other direction. Consider any feature F and $v \in M_F$, there

are bundles $\textsc{ConfBalls}_F(v)$ and $\textsc{RepBalls}_F(v)$ representing all the Balls intersecting $V_{M_F}(v)$. The union of these two bundles is denoted $\textsc{Balls}_F(v)$. There is another bundle $\textsc{Collars}_F(v)$ for the Collars intersecting v's Voronoi cell.

The low-level access to the BLG is given by AddLink and RemoveLink (Figure 3.3). RemoveLink merely deletes the obvious pointers. When AddLink is called, it is to register new intersections between balls and Voronoi cells. Such intersections are the cause of collar and encroached events, so AddLink traps such events and enqueues them if necessary.

Higher-level access to the BLG is given by UpdateLocation and PropagateLocation (Figures 3.3 and 3.3). UpdateLocation is called with the old and new vertex sets when a mesh M_F has been changed. In this case a few things happen, first, any local RepBalls must be recomputed since the restricted Delaunay may have changed. Then all the location objects must be re-intersected with the new Voronoi cells. New ConfBalls must be created to send to higher dimension. At this point ConfBalls should not be created around *scaffolding*, simplices inserted into the lower-dimensional Ω_F that are do not contain part of F. This cruft is not should stay invisible to higher-dimensional meshes. The last step in UpdateLocation is to send the all the revised location objects to PropagateLocation to inform the higher dimensional meshes.

PropagateLocation is passed an old set and a new set of location objects and a feature. All the superfeatures must be updated, removing links to the old objects and adding links to the new objects. The superfeature may not be conforming to all the new ConfBalls, this is the cause of unresolved events, and so it is trapped here and events are enqueued as necessary. Inside the collar region, subfeatures need not be resolved, so no new events for these are created.

3.3 Detailed Algorithm

SVRC($\mathcal{P}$: a piecewise linear complex, τ, θ, σ)
1: INITIALIZE($\mathcal{P}$)
2: $B :=$ DEQUEUE()
3: **while** $B \neq \emptyset$ **do**
4: DESTROY(B)
5: $B :=$ DEQUEUE()
6: **end while**
7: **return** RIPSTITCHCOLLAR()

Figure 3.2: Main method for `SVRC`. Initialization creates meshes for each feature and links them together in the BLG. `SVRC` repeatedly DESTROYS gap balls to achieve conforming and quality goals. DEQUEUE chooses balls in priority order to ensure good runtime. Finally, the call to RIPSTITCHCOLLAR removes whatever cruft is inside the collar system and replaces it with a no-large-angle conforming tetrahedralization.

INITIALIZE($\mathcal{P}$: a piecewise linear complex)
1: Compute CREASES, the creases of $\mathcal{P}$
2: Create $M_\Omega :=$ BOUNDINGBOX($\mathcal{P}$)
3: **for** $F \in \mathcal{P}$ **do**
4: Create $M_F :=$ BOUNDINGBOX(F)
5: **end for**
6: **for** $F \in \mathcal{P}$ **do**
7: UPDATELOCATION($\emptyset, M_F, F$)
8: **end for**

Figure 3.3: Initialization begins by building a constant sized bounding mesh for each feature. UPDATELOCATION creates new CONFBALLS for a feature and sends them to superfeatures, which will enqueue new work events to initially load the queues.

DESTROY($B = \langle (c, r), F \rangle$: a BALL)

1: **if** c is no longer on the Voronoi skeleton of M_F **then return**
2: Compute $\mathcal{V} := \{v \in M_F \mid V_{M_F}(v) \cap B \neq \emptyset\}$: the set of Voronoi cells intersecting B
3: Compute $\mathcal{B} = \bigcup_{v \in \mathcal{V}} \text{BALLS}_F(v)$
4: Compute $\mathcal{W} = \{\langle (p, 0), F' \rangle \in \mathcal{B} \mid p \in B(c, \theta r) \text{ and } \text{CD}(B) < \text{CD}(p)\}$: warp points
5: **if** $\exists\ \langle (p, 0), F') \rangle \in \mathcal{W}$ **then**
6: Choose p to minimize $\text{CD}(p)$
7: FORCEINSERT(F, p, F')
8: **else**
9: Compute $\mathcal{Y} := \{B' \in \mathcal{B} \mid c \in B'^{\circ} \text{ and } \text{CD}(B') < \text{CD}(B)\}$:yield balls
10: **if** $\mathcal{Y} = \emptyset$ **then**
11: FORCEINSERT(F, c, F)
12: **else**
13: Take $B' \in \mathcal{Y}$, DESTROY(B')
14: **goto** 1
15: **end if**
16: **end if**

Figure 3.4: The central operation of `SVRC` is to continually DESTROY large gap balls. If the center of the ball is no longer on the Voronoi skeleton, it has been destroyed by other operations. If there is an unresolved vertex p (BALL with radius 0) within θr of c, then c should warp to p conforming purposes(line 7). In line 6, `SVRC` carefully inserts the candidate of lowest containment dimension. This ensures that points on creases are added before too many points are added near a crease, so that collars can be created to stop refinement. Next, DESTROY then looks for any nearby lower-dimensional protective balls encroached by c. If there aren't any encroached balls, it is safe to insert c (line 11). If there is some encroached ball B', this call to DESTROY is a witness that B' must be destroyed, so yield to destroying B'. The destruction of B' may have created some new points to warp to so go back and check everything again (line 14). The computations of $\mathcal{V}$, $\mathcal{B}$, $\mathcal{W}$ and $\mathcal{Y}$ must be cached so that the conditionals are fast on repeated attempts.

ForceInsert(F: feature to add to, p: a point, F': feature containing p)

1: Compute $\mathcal{V} := \{v \in M_F \mid V_{M_F}(v) \neq V_{M_F \cup \{p\}}(v)\}$: the set of vertices whose Voronoi cells that will change
2: Set $\mathcal{V}' := \mathcal{V} \cup \{p\}$.
3: Add p to M_F, creating new cells $V_{M_F}(v')$ for every $v' \in \mathcal{V}'$
4: UpdateLocation($\mathcal{V}, \mathcal{V}', F$)
5: **if** $dim(F') < dim(F)$ and $p \in$ Creases **then**
6: CreateCollar(p, F)
7: **end if**
8: **for all** $v' \in \mathcal{V}'$ **do**
9: **if** v' is the center of some work event O on a work queue Queue **then**
10: DecreaseSize(O, Queue, $R_{v'}$)
11: **end if**
12: **if** v' is a τ-skinny Voronoi cell **then**
13: let c be the farthest Voronoi corner of $V(v')$
14: Enqueue($\langle (c, R_{v'}), F \rangle$, skinny, F)
15: **end if**
16: **end for**

Figure 3.5: The ForceInsert routine always inserts a point p into a mesh of a feature F. Update the mesh M_F, then update the BLG with UpdateLocation. If p is on a crease, augment the collar system to protect p. Lastly, enqueue any new skinny events for destruction.

UpdateLocation(V: old vertices, V': new vertices, F:containing feature)
1: Compute $\mathcal{C} := \cup_{v \in \mathcal{V}} \text{Collars}_F(v)$ the old Collars
2: Initialize $\mathcal{S}' = \emptyset$ to collect new ConfBalls
3: Initialize $\mathcal{R}' = \emptyset$ to collect new RepBalls
4: **for all** $v' \in \mathcal{V}'$ **do**
5: **for all** $C \in \mathcal{C}$ **do**
6: Create RepBalls for any intersections in $C \cap V_{M_F}(v')$, add these to $\mathcal{R}'$
7: **if** $|v'C| \leq \delta R(C)$ **then**
8: Snap(v',C)
9: **end if**
10: **end for**
11: **for all** Simplices $T \in \text{Del}(M_F)$ with v' as a vertex **do**
12: **if** $|T| = dim(F)$ or $|T| = 1$ **then**
13: Let $B(c, r)$ be the diametral ball of T
14: **if** $dim(T \cap F) = dim(T)$ **then** create ConfBall$\langle (c, r), F \rangle$, add to $\mathcal{S}'$
15: **end if**
16: **end for**
17: **end for**
18: Compute $\mathcal{S} := \cup_{v \in \mathcal{V}} \text{ConfBalls}_F(v)$ the old lower dimensional ConfBalls
19: Compute $\mathcal{R} := \cup_{v \in \mathcal{V}} \text{RepBalls}_F(v)$ the old RepBalls
20: Set $\mathcal{O} := \mathcal{C} \cup \mathcal{S} \cup \mathcal{R}$, all the old location objects
21: Set $\mathcal{O}' := \mathcal{C} \cup \mathcal{S} \cup \mathcal{R}'$, the location objects to re-intersect
22: **for all** $v \in \mathcal{V}$, RemoveLink$(v, \mathcal{O})$)
23: **for all** $v' \in \mathcal{V}'$, $\{O' \in \mathcal{O}' |\ O' \cap V_{M_F}(v') \neq \emptyset\}$ **do**
24: AddLink$(v',\ F,\ O')$
25: **end for**
26: Set $\mathcal{O}'' := \mathcal{S}' \cup \mathcal{R}'$, the new objects to report to higher dimension
27: PropagateLocation$(\mathcal{O},\ \mathcal{O}'',\ F)$

Figure 3.6: UpdateLocation performs local operations to update the BLG when a patch of M_F has been updated. First, create new RepBalls to account for the new restricted Delaunay, then create new ConfBalls to report to higher dimension. Some of the new ConfBalls may only be bounding-box scaffolding in M_F around F. Superfeatures need not resolve these and should not be informed, so discard scaffolding here by checking in line 14. Re-intersect the new Voronoi with the old collars, new RepBalls and old lower-dimensional ConfBalls. Lastly, propagate the new location objects to higher dimension.

CreateCollar(p: point on crease, F: feature needing collar)

Compute $g :=$ FastCollarSize(p, F)
Compute $\mathcal{V} := \{v \in M_F \mid V(v) \cap B(p, g) \neq \emptyset\}$ covering the new collar ball
Compute $\mathcal{C} := \bigcup_{v \in \mathcal{V}}$ Collars$_F(v)$ the old Collars
Compute $\mathcal{R} := \bigcup_{v \in \mathcal{V}}$ RepBalls$_F(v)$ the old RepBalls
Compute $\mathcal{C}'$, augmenting collar system $\mathcal{C}$ to include $B(p, g)$
Initialize $\mathcal{R}' = \emptyset$ to collect new RepBalls
for all $v \in \mathcal{V},\ C \in \mathcal{C}'$ **do**
    Create any RepBalls due to intersections $C \cap V_{M_F}(v)$, add these to $\mathcal{R}'$
    For any segments emanating from a vertex collar, enqueue the dimension (1,2) vertex as Unresolved
end for
for all $v \in \mathcal{V}$ **do**
    RemoveLink$(v, \mathcal{C} \cup \mathcal{R}))$
    for all $O \in \mathcal{R}' \cup \mathcal{C}' \mid O \cap V_{M_F}(v) \neq 0$ **do**
        AddLink$(v,\ F,\ O)$
    end for
end for
PropagateLocation$(\mathcal{C} \cup \mathcal{R},\ \mathcal{C}' \cup \mathcal{R}',\ F)$

Figure 3.7: CreateCollar first calls FastCollarSize to determine the size of the collar ball to be created. Neighboring portions of the collar system are found to update the collar surface. New RepBalls are created. The Collars and RepBalls in the BLG must then be updated. and this information propagated to higher dimension.

FastCollarSize(v: vertex on crease, F: feature needing collar) $\rightarrow \mathbb{R}$

Initialize $\mathcal{F} = \emptyset$, the set of features near v
Initialize $g = \infty$, the size of the neighborhood needing to be explored
Initialize $X = \emptyset$, the region that has been explored
while $B(v, g) \not\subset X$ iterate on $v' \in M_F$ in breadth-first order beginning with v **do**
    Update $X := X \cup V_{M_F}(v')$
    Update $\mathcal{F} := \mathcal{F} \cup \{F' \mid \exists\ \langle (c, r), F' \rangle \in$ ConfBalls(v') and $F' \neq F\}$
    Update $g := \min\{|vF'| \mid F' \in \mathcal{F}\}$
end while
Return $g/3$

Figure 3.8: FastCollarSize computes the size of a proper collar ball around v. A neighborhood of v is explored. g keeps track of the nearest feature that has been found. Loop until there cannot be a feature nearer to v. When the loop terminates, $g = g(v)$. The actually procedure is rather brute force, with the cleverness in its analysis (Chapter 5).

```
PropagateLocation(O, O', F)
 1: for all superfeatures F' ∈ {F' ∈ P | F' ⊃ F} do
 2:   Compute V := ⋃_{O∈O} (Cells_{F'}(O)), cells in F' needing to be updated
 3:   for all v ∈ V do
 4:     RemoveLink(v, F', O)
 5:     for all O' ∈ O' do
 6:       if V_{M_{F'}}(v) ∩ O' ≠ ∅, then AddLink(v, F', O')
 7:     end for
 8:   end for
 9:   if O' = ⟨(c, r), F⟩ is a ConfBall and F ∩ O' ≠ F' ∩ O' then
10:     Let C = Collars_{F'}(v): collar balls
11:     if V_{F'}(v) ⊄ Collars_{F'}(v) then
12:       Enqueue(O', unresolved, F')
13:       if v is on a crease, isolated, and θ0-medial then
14:         CreateCollar(v,F')
15:       end if
16:     else
17:       Let c = creases(C): center of collar ball containing v
18:       if c ∈ creases AND CD(c') = 1 then
19:         Enqueue(O', unresolved, F')
20:       end if
21:     end if
22:   end if
23: end for
```

Figure 3.9: After a mesh inserts a point into F, all superfeatures F' need to be informed about the new set of location objects. Loop over superfeatures, updating the BLG for each one. New ConfBalls may add new items to the unresolved queue to make a superfeature F' conform. If this occurs internal to the collar system (line 11)
and is not on the interior of a segment crease, then F' does not have to conform, so PropagateLocation does not add work event. If this lower dimensional point is on a crease, it may be time to augment the collar system.

```
AddLink(v:vertex, F:containing feature, O: new location object)
 1: Add v to Cells_F(O)
 2: if O is a Collar then
 3:    Add O to Collars_F(v)
 4: else if O is a RepBall then
 5:    Add O to RepBalls_F(v)
 6:    if O has bad topology or the representation angle is worse than σ then
 7:       Enqueue(O, collar, F)
 8:    end if
 9: else
10:    Add O to ConfBalls(v)
11:    if v encroaches O then
12:       if CD(v) = dim(F) then
13:          Enqueue(O, encroached, F)
14:       else if PartiallyEncroached(O) then
15:          Enqueue(O, encroached, F)
16:       else
17:          PartiallyEncroached(O) := True
18:       end if
19:    end if
20: end if
```

```
RemoveLink(v:vertex, F:containing feature, 𝒪: old location objects)
1: for all O ∈ 𝒪 do
2:    Remove O from ConfBalls_F(v), RepBalls_F(v), and/or Collars(b) appropriately
3:    Remove v from Cells_F(B)
4: end for
```

Figure 3.10: AddLink updates the BLG. New RepBalls may be enqueued for destruction if they have bad representation angle or bad topology. New ConfBalls may require destruction if singly-encroached by a higher dimensional vertex or if doubly-encroached. RemoveLink makes a simple set removal from the BLG.

```
ENQUEUE(B : Ball, TYPE : aworkeventtype, F : Feature) → BALL
 1: PUSH(TYPEQUEUE_{dim(F)}, B)

DEQUEUE() → BALL
 1: for i = 1 to 3 do
 2:    if SKINNYQUEUE_i ≠ ∅ then return  POP(SKINNYQUEUE)
 3: end for
 4: for i = 2 downto 1 do
 5:    if ENCROACHEDQUEUE_i ≠ ∅ then return  POP(ENCROACHEDQUEUE)
 6: end for
 7: for i = 3 downto 1 do
 8:    if Q(UNRESOLVED, i) ≠ ∅ then
 9:       B = ⟨(c, r), F⟩POP(UNRESOLVEDQUEUE)
10:       for all F' ∈ P | F' ⊃ F do
11:          if F ∩ B ≠ F' ∩ B then return  B
12:       end for
13:    end if
14: end for
15: for i = 2 to 3 do
16:    if COLLARQUEUE_i ≠ ∅ then return  POP(COLLARQUEUE)
17: end for
18:  return ∅
```

Figure 3.11: ENQUEUE merely marshalls work into the appropriate dimensional queue. DEQUEUE accesses the work queues in priority order. Skinny, encroached, and collar angle balls will need to be destroyed unless they have already been destroyed by some other insertion (checked later by DESTROY). However, unresolved elements may have become resolved. Trapping this behavior in DEQUEUE allows DESTROY to perform ignorantly of the type of work event.

RipStitchCollar() → A Triangulation of Ω

1: Set $\mathcal{T}$:= the Delaunay diagram dual to M_Ω
2: Compute $\mathcal{C} := \cup_{v \in M_\Omega} \textsc{Collars}(v)$
3: Compute $\mathcal{V} := \{v \in M_\Omega \mid v \in \textsc{Creases}\}$
4: Compute $\mathcal{I} := \{v \in M_\Omega \mid V(v) \subset \mathcal{C}\}$
5: Remove from $\mathcal{T}$ all simplices adjacent to $\mathcal{I}$.
6: Initialize $\mathcal{P} = \emptyset$, a set of pairs
7: **for all** $v \in \mathcal{V}$ **do**
8: **if** there is a sphere $S := \{S \in \textsc{Collars}(v) \mid C = B(v, \_)\}$ **then**
9: Add vertex v to $\mathcal{T}$
10: **end if**
11: **if** CD$(v) = 1$ **then**
12: **for all** $C \in \textsc{Collars}(v) \mid \exists\ v' \in \mathcal{V},\ C = B(v, \_) \cap B(v', \_)$ (there are 2) **do**
13: Add $\langle \{v, v'\}, C \rangle$ to $\mathcal{P}$
14: **end for**
15: **end if**
16: **end for**
17: **for all** $\langle T, C \rangle \in \mathcal{P}$ **do**
18: Compute $\mathcal{R} := \bigcup_{m \in \textsc{Cells}_\Omega(C)} \textsc{RepBalls}(m)$
19: **for all** $B \in calR$ **do**
20: Let T' be the restricted Delaunay simplex dual to the Voronoi polytope containing B's center
21: Add Sim$(T' \cup T)$ and its subsimplices to $\mathcal{T}$
22: **end for**
23: **end for**
24: **return** $\mathcal{T}$

Figure 3.12: RipStitchCollar performs a post process on the volume mesh M_Ω to replace the triangulation inside the collar system. First compute the collar region $\mathcal{C}$, the crease vertices $\mathcal{V}$, and the vertices internal to the collar region $\mathcal{I}$. Empty out the collar regions in line 5. The first loop then pairs up crease points with collar spheres, and the inner loop pairs crease edges with collar circles. The second loop goes over all the pairs and fills $\mathcal{T}$ with a star around each crease point and a book around each crease edge.

Chapter 4

Termination and Sizing Analysis

The goal of this chapter is to prove two properties of the algorithm SVRC using just one theorem. The Spacing Theorem lowerbounds the nearest neighbor as a nonzero function on the input $\mathcal{P}$. Since this sizing is nonzero, packing gives an upper bound on the number of vertices that will fit, thus proving the termination of SVRC.

Many are extraneous vertices are inserted by SVRC that are not included in the final output. The other use of the Spacing Theorem is to prove the Scaffold-Sizing Theorem in the next Chapter, guaranteeing these extraneous vertices do not cause significant work.

4.1 Spacing

Recall from Section 1.4, that for a PLC $\mathcal{P}$ without creases, an size-optimal, conforming, well-spaced point set M has $f_M \gtrsim f_{\mathcal{P}}$. For handling creases, no optimal bounds or algorithms are known. SVRC will still bound its spacing as a function of the input. Given a PLC $\mathcal{P}$, recall that the collar region $\hat{\mathcal{C}}$ is the area around creases with sizing based on $g_{\mathcal{P}}$. This chapter will define $\hat{\mathcal{C}}$ as the ε_2-collar-region, where ε_2 is a constant defined in Lemma 24. Recall that the clipped complex $\mathcal{P}_{\hat{\mathcal{C}}}$ is is the union of features with portions removed inside the collar region. The goal is to lower bound the spacing in SVRC by the function $f_{\mathcal{P}_{\hat{\mathcal{C}}}}$. To mark its importance call this function the **input spacing** and denote it $\Psi := f_{\mathcal{P}_{\hat{\mathcal{C}}}}$.

The algorithm is only ever approximately aware of the collar region $\hat{C}$, so at any time there is a approximate collar region $\mathcal{C}_i$, these increase as more collars are inserted, to form a final approximation $\mathcal{C} := \cup_i \mathcal{C}_i$. Since the approximation is made of balls with radius $g/3$, it will be good enough to cover the ε_2-collar-region $\hat{\mathcal{C}}$.

For this chapter, consider each feature mesh M_F as a sequence of meshes M_F^i taken over the run of the algorithm. Whenever F or i is suppressed, the statements are to hold for any choice of F and i. Sequences are nested so that $M^i \subset M^{i+1}$. Meshes may be denoted M^- and M^+ to refer to the mesh immediately before and after an insertion.

To prove termination, first I show in Section 4.1.1 a weak condition that the spacing in SVRC is lower-bounded by a spacing function that goes to zero around the creases. Nonetheless, this will guarantee that SVRC terminates in the area outside the creases. Then, I show that because refinement is stopped inside the creases (Section 4.1.3), the spacing in the creases is lower bounded by the non-zero spacing in the exterior (Section 4.1.4). This will guarantee termination.

A few technical results are also contained in this chapter are useful for both termination and later for runtime. In particular, Section 4.1.2 gives the proof that every mesh has at least some quality bound throughout the run of the algorithm.

4.1.1 Weak Spacing

Define a new spacing function $\bar{g}$ as follows:

Definition 39 (Lipschitz $\bar{g}$)**.** *Given a set of features $\mathcal{X}$, define the function $\bar{g}_{\mathcal{X}}$ as the largest function such that $\bar{g}_{\mathcal{X}} \leq g_{\mathcal{X}}$ and $\bar{g}$ is 1-Lipschitz.*

The function $\bar{g}$ is called the weak spacing function. Note that the definition of $\bar{g}$ is equivalent to definition $\bar{g}$ as the distance to the second nearest feature, so that $\bar{g} < f$ and $\bar{g} < g$. The weak spacing function approaches zero at all the creases, and essentially captures the sizing of the mesh if a Delaunay refinement algorithm that didn't properly handle creases were to run forever. A mesh that is sized according weak spacing will be infinite, but it can still conform to the input and have good quality. This notion of an *infinite mesh* simplifies the analysis and allows the interior of the collar regions to be subjected to a seperate analysis. Weak spacing on it's own is powerful enough to give the desired runtime bound if the algorithm, albeit with an infinite M, in the sense that any partial run of the algorithm will conform to the output-sensitive runtime bound for the partial output.

Several of the cases within the proof of the Weak Spacing Theorem are purely inductive in nature, depending only on the Lipschitz condition. These may be recalled in later proofs relative to other Lipschitz sizing functions.

Theorem 5 (Weak Spacing Theorem)**.** *There exists $K \in (0, \infty)$, such that for any point p considered for insertion into any mesh M^i:*

$$\bar{g}(p) \leq K n_{M^i}(p)$$

Proof. If Theorem 5 holds, there is an immediate corollary that bounds the spacing across sequences:

Corollary 2. *In any mesh M, for any $v \in M$, we have that:*

$$\bar{g}(v) \leq (1 + K) n_M(v)$$

Proof. The vertex v inserted at some point into some mesh M^i, so by Theorem 5, $\bar{g}(v) \leq Cn_{M^i}(v)$. Let w be the nearest neighbor of v in M, so $n_M(v) = |vw|$. Vertex w was inserted at some point j, so $\bar{g}(w) \leq Kn_{M^j}(w)$ If $j < i$, then $w \in M^i$ so the corollary is trivially complete:

$$\bar{g}(v) \leq Kn_{M_i}(v) \leq K|vw| = Kn_M(v)$$

If $i < j$, then by a symmetric argument $\bar{g}(w) \leq Kn_M(w)$, so then by Lipschitz:

$$\bar{g}(v) \leq |vw| + \bar{g}(w) \leq |vw| + Kn_M(w) \leq |vw| + K|vw| = (1+K)|vw| = (1+K)n_M(v)$$

□

The proof of the Weak Spacing Theorem will proceed inductively on a sequence of insertions. Note that the if the lemma holds for all i less then some i_0, then the corollary also holds for $i < i_0$. Thus the Corollary2 may be used as an inductive hypothesis in proving Theorem 5 Note the corollary trick was independent of $\bar{g}$ except that it was Lipschitz.

Inductively assume the lemma holds not for a single constant K, but for a bundle of constants K^i_j for every pair of containment dimensions with $j < i$. Constant K^i_j will govern the sizing for a vertex of containment dimension j added into a mesh of dimension i. Take $K^i = \max_j K^i_j$. Then when any vertex v is inserted into a mesh M of dimension i:

$$\bar{g}(v) \leq K^i n_M(v)$$

Similarly, take $K_j = max_i\{K^i_j\}$, then a vertex v of containment dimension j inserted into *any* mesh M has:

$$\bar{g}(v) \leq K_j n_M(v)$$

A complicated set of constraints will be derived on the bundle of constants in order to satisfy the lemma, then these will be shown to be feasible. Setting $K = \max_i\{K^i\}$ will finish the lemma.

A point p is considered for one of four reasons: Skinny, Encroached, Unresolved, and Collar. The inductions steps in the proof are *independent* of the definition of $\bar{g}$ except that it is Lipschitz. These cases will be recalled for brevity in later proofs.

Case 1: Skinny

Suppose p is considered for M_F for reason Skinny, let $i := dim(F) = \text{CD}(p)$. Now p is the farthest Voronoi corner of some skinny cell $V(v)$. $V(v)$ is skinny, so $R_v/r_v > \tau$. Consider then:

$$\bar{g}(p) \leq |pv| + \bar{g}(v) \leq R_v + (1+K^i)n_M(v) = R_v(2+2K^i)r_v \leq (1+\frac{2+2K^i}{\tau})R_v = (1+\frac{2+2K^i}{\tau})n_M(p)$$

Then $K^i_i n_M(p)$ must be larger than the right hand side. This not satisfiable unless $\boxed{\tau > 2}$, and the general constraint $\boxed{K^i_i > (1 + \frac{2+2K_i}{\tau}) \text{ for } i = 1, 2, 3}$

Case 2:UNRESOLVED

There are two subcases, depending on how UNRESOLVED event is carried out, either warping to an unresolved point, or inserting a Steiner point because the unresolved point is not $(1-\theta)$-medial.

The first case 2(A) is when `SVRC` warps to an unresolved point.
Suppose a mesh M of dimension i inserts some UNRESOLVED point p of dimension j, so that $j \leq i-1$. Let q be the nearest-neighbor of p in M.
There are three sub-sub-cases based on the containment dimension of q and its relation to p.
Case 2(A)(i): Suppose $CD(q) = i' > i-1$. Then q was inserted as the center of some gap-ball of M with radius R. Since the insertion of q did not warp to p, it follows $|pq| > \theta R$. Continuing:

$$\bar{g}(p) \leq \bar{g}(q) + |pq| \leq K^i_{i'}R + |pq| < (1 + \frac{K^i_{i'}}{\theta})|pq| = (1 + \frac{K^i_{i'}}{\theta})n(p)$$

The right-hand side must be less than $K^i_j n(p)$, so this leads to the set of constraints:

$$\boxed{K^i_j > 1 + \frac{K^i_{i'}}{\theta} \text{ for } i = 1,2,3,\ j \leq i-1 \text{ and } i' > i-1}$$

Case 2(A)(ii): Suppose $CD(q) \leq i-1$ and q and p have different parent features. This is a base case, since $\bar{g}(q) \leq g(q) \leq 2|pq|$, this is trivially satisfied by

$$\boxed{K^i_j > 2 \text{ for } i = 1,2,3 \text{ and } j \leq i-1}$$

Case 2(A)(iii): Then $CD(q) \leq i-1$ and q and p must have the same parent feature F, with $j := dim(F) = \text{CD}(q) = \text{CD}(p)$. By simple induction, then $\bar{g}(q) \leq (1 + K^j_j)|pq|$, this is satisfied by

$$\boxed{K^i_j > 1 + K^j_j \text{for } i = 2,3 \text{ and } j \leq i-1}$$

Case 2(B): The last case for UNRESOLVED is when `SVRC` inserts a Voronoi corner p due to an unresolved point q that is not $(1-\theta)$-medial. Suppose $CD(q) = j$, and suppose p is a corner of the $V(v)$ with $q \in V(v)$. Since q is not $(1-\theta)$-medial, then by definition, $|vq| \leq 2(1-\theta)R_v/\theta$. Point q will eventually be inserted, so by induction $\bar{g}(q) \leq K^i_j|vq|$. Then consider:

$$\bar{g}(p) \leq |pq| + \bar{g}(q) \leq 2R_v + K^i_j|vq| \leq 2(1 + \frac{1-\theta}{\theta}K^i_j)R_v$$

So this adds the constraint:

$$\boxed{K^i_i > 2(1 + \frac{1-\theta}{\theta}K^i_j) \text{ for } i = 1,2,3 \text{ and } j \leq i-1}$$

Case 3:ENCROACHED

Take $B := B(p, R)$ as the ball that is encroached, and let v be one of the vertices on B. Let $i := \mathrm{CD}(p)$. Let q be the point that encroached on B, let $j = \mathrm{CD}(q)$. There are two sub-cases, depending on j.

Case 3(A): Suppose $j \leq i$. Then q is not on the same feature as p, so this is a simple base case with $\bar{g}(p) \leq g(p) \leq R = n(p)$, satisfied by the constraint

$$\boxed{K_i^i > 1 \text{ for } i = 1, 2}$$

Case 3(B): Then $j > i$ and q is on a superfeature. Consider the j'-dimensional mesh M' containing q. When q was inserted into M', it was the center of a gap-ball of radius R_q not containing v and q did not warp to v, so $|qv| > \theta R_q$. Then:

$$\bar{g}(p) \leq |pq| + \bar{g}(q) \leq R_v + K_j^{j'} R_q < R_v + |qv| \frac{K_j^{j'}}{\theta} \tag{4.1.1}$$

$$\leq R_v + (|vp| + |pq|) \frac{K_j^{j'}}{\theta} = (1 + \frac{2K_j^{j'}}{\theta}) R_v = (1 + \frac{2K_j^{j'}}{\theta}) n_M(p) \tag{4.1.2}$$

When p is the midpoint of a straight-line segment, a better bound of $|vq| \leq \sqrt{2} R_v$ is achieved by using both ends. This yields the bundle of constraints (including encroached representation balls):

$$\boxed{K_{(3,2)}^3 > 1 + \frac{2}{\theta} K_3^3}$$

$$\boxed{K_2^2 > 1 + \frac{2}{\theta} K_3^3}$$

$$\boxed{K_1^1 > 1 + \frac{\sqrt{2}}{\theta} K_3^3}$$

$$\boxed{K_1^1 > 1 + \frac{\sqrt{2}}{\theta} K_2^2}$$

$$\boxed{K_{(2,1)}^2 > 1 + \frac{2}{\theta} K_3^3}$$

$$\boxed{K_{(2,1)}^2 > 1 + \frac{2}{\theta} K_2^2}$$

$$\boxed{K_{(2,1)}^2 > 1 + \frac{2}{\theta} K_{(3,2)}^3}$$

Note, the last three constraints are the encroachment of a REPBALL protecting an arc segment where a collar sphere intersects an input facet. This bound can be improved since the arc segment approaches a straight line in the limit.

Suppose p is the center of an arc-segment of a collar sphere centered at c. Then the radius R_c of the sphere is $g(c)/3$, so $\bar{g}(p) \leq g(p) \leq R_c + g(c) = 4R_c$. Let a, b be the endpoints of the arc-segment, and let a' and b' be the corresponding endpoints of the straight line segment through p tangent to the collar. Let some ε close to zero. When the RepBall has radius $R_p \leq \varepsilon R_c$, then $|aa'|, |bb'| \lesssim \varepsilon$, so better encroachment bound gives new constraints of the form:

$$\boxed{K^2_{(2,1)} > 1 + \frac{\sqrt{2} + O(\varepsilon)}{\theta} K^3_3}$$

Suppose the RepBall has radius $R_p \geq \varepsilon R_c$, then re-class this as a base-case, i.e.:

$$\bar{g}(p) \leq 4R_c \leq \frac{1}{\varepsilon} R_p = \frac{1}{\varepsilon} n(p)$$

So we have:

$$\boxed{K^2_{(2,1)}} > \frac{1}{\varepsilon}$$

Replacing each of the earlier constraints with this pair will give a system that is feasible for better choices of τ. Note that ε is only realized in the analysis.

Case 4: Collar

The Collar case requires the follow Lemma:

Lemma 17. *`SVRC` generates a smooth collar system or terminates larger than $\bar{g}$.*

Proof. Consider two collar balls created by `SVRC` that intersect. First, suppose they are both centered on the same segment and neither is an endpoint. Simple geometry shows that if the two collar balls intersect at sharper than a right angle, then any points on the intersection would encroach upon the segment. If no points on the intersection are resolved, then the Voronoi cells coverting the vertices of the segment must be larger than $g/4$ which is larger than $\bar{g}$.

Consider where a collar ball $B := B(v, R)$ at a crease vertex intersects a collar ball centered at u on some crease segment adjacent to v. First, it must be that $|uB| > \delta R$, otherwise B would warp. But in this case, the appropriate vertex with containment dimension $(1, 2)$ would have been added, ensuring the smoothness. □

Case 4(A): Suppose all the vertices of the bad simplex are on the collar, this is a base case. Here, p is point on the collar surface with bad representation angle or bad topology. Let c and R_C be the center and radius of the piece of the collar description to which p corresponds, recall $R_C = g(c)/3$. ε-sampling theory (Lemma 16) guarantees that the since the representation ball around p is bad, it must be big relative to R_C, i.e. there exists ε such that $n(p) \geq \varepsilon R_C$. Then

$$\bar{g}(p) \leq |pc| + \bar{g}(c) \leq R_C + g(c) = 4/3 R_C \leq 4/3 \varepsilon n(p)$$

. So the constraint $\boxed{K^i > 4/3\varepsilon \text{ for } i = 2, 3}$ suffices. Note that the only requirement on $\bar{g}$ for this case was that $\bar{g}(c) \leq g(c)$ for points on a crease.

Case 4(B): Suppose some COLLAR event is inserting p, and suppose the bad simplex has at least two vertices off the collar, let them be v and w. The two vertices v and w both have higher containment dimension then p, and they are nearby, so the insertion of p can be charged inductively to the distance $|vw|$. The resulting constraints and derivation are identical to those as if the REPBALL had been encroached as in Case 3.

Case 4(C): Inserting p, suppose all but one vertex v of the bad simplex is on the collar. This will be a base case. If v had snapped to the collar, then all the vertices would be symbolically on the collar, and so the simplex would be good. Thus the vertex v did not snap to the collar, so the distance from v to the collar is at least δ times the radius R of the collar ball. From Lipschitz and since $R = g/3$ at the collar center, then $R \geq g(p)/4$. Thus $|vp| \geq \delta g(p)/4$, so that:

$$\bar{g}(p) \leq g(p) \leq \frac{4}{\delta}|vp| = \frac{4}{\delta}n(p)$$

So

$$\boxed{K > \frac{4}{\delta}}$$

will suffice. □

Feasibility for Sizing Constants

The constraints on the family of constants K in the weak spacing proof are all of the form $K > a + bK'$ for some K and K'. So these form a linear program, with the additional set of constraints that all the K must be strictly greater than zero. If this program is feasible, then there exists a family of constants to satisfy all the induction hypothesis and thus prove the theorem.

I claim the set of constraints is feasible whenever $\tau\theta^2 > 4\sqrt{2}$. Since $\theta < 1$, this implies $\tau > 4\sqrt{2}$. Since the program is finite, this can be easily verified by computer algebra.

It is useful to expose a more generic technique. The first claim is that because all the constants must be positive, then by scaling arguments, every constraint $K > a + bK'$ can be replaced by a constraint $K > bK'$. If the latter system is feasible, then a solution to the latter can be multiplied by some large constant to give a solution to the former.

To question the feasibility of this simpler system, create a graph with a node for every K. For every constraint $K > bK'$, add a directed edge from K to K' with weight b. Some of the constraints may form self-loops, and it may be a multigraph. (The multigraph case can be reduced by throwing out dominated constraints.) The system is then feasible if the product of the weights along any cycle is strictly less than 1. For the family of constraints K, the critical cycle will have weight $\tau\theta^2/4\sqrt{2}$, leading to the constraints on τ and θ for feasibility.

Shewchuk has alluded to this technique with the use of "data-flow diagrams" [She97b]. A rigorous treatment is given in Section 24.4 of [CLRS01]. The latter treatment is framed using additive constraints and summing weights over cycles. The system here can be reduced to additive constraints by taking the log of each constraint to get $\log K > \log b + \log K'$; it is clear that feasibility is preserved by this reduction.

4.1.2 Mesh Quality Maintenance

A key property on which many bounds depend is that at every stage of the algorithm, the mesh is always a quality mesh. This determines the runtime and space bounds, and is also needed to ensure correctness of the dynamic collar process. The quality at intermediate stages is not the final τ quality bound that the user calls for – `SVRC` cannot guarantee that the quality won't degrade by some amount – but is only smaller by a constant function of parameters τ and θ.

The idea is to show that `SVRC` decays the mesh sizing gradually enough that quality is never truly lost. To this end, I begin with a few technical lemmas bounding the reduction in sizing as refinement occurs.

Lemma 18 (Any Single Insertion Only Marginally Decreases Quality)**.** *Suppose `SVRC` inserts a single vertex into a τ-quality mesh, then there exists τ_1 depending only on τ and θ such that M' is a τ_1-quality mesh.*

Proof. The first observation is that every vertex inserted by `SVRC` is $(1-\theta)$-medial with respect to the previous point set. Any centers of protective balls are 1-medial, and by design, `SVRC` only warps to unresolved vertices that are $(1-\theta)$-medial.

Since every insertion is $(1-\theta)$-medial, the totality of `SVRC` can be viewed as a well-paced extension (Section 2.3) of the initial bounding box. Applying Lemma 12, it follows that for any vertex $v \in M$:

$$f_M(v) \lesssim f_{M'}(v) \tag{4.1.3}$$

Consider also the new vertex u. Firstly $f_M(u) \leq 2R_p$, where p is chosen so that $u \in V_M(p)$. Also $n_{M'}(u) \geq (1-\theta)R_p$, from which it follows that $n_{M'}(u) \geq \frac{1-\theta}{2} f_M(v)$. Since $n_{M'}(u) = f_{M'}(u)$, then equation 4.1.3 also holds for the new vertex, and thus all vertices of M'.

Recall Section 2.2.3 on grading for the rest of the argument. For every vertex v in M', clearly $G_{M'}(v) \leq G_M(v)$, since refinement can only decrease the gap-size G.

Since the gap-ratio Γ is the ratio of G to f, it follows immediately that for all vertices $v \in M'$:

$$\Gamma_{M'}(v) \lesssim \Gamma_M(v)$$

Since M was τ-quality, by Lemma 8 I then obtain that for every vertex $v \in M'$:

$$\Gamma'_M(v) \lesssim \Gamma_M(u) \lesssim 1$$

Since $\Gamma_{M'}$ is bounded at every vertex, it then follows from Lemma 9 that M' is a τ_1-quality mesh where $\tau_1 \lesssim \tau$. (Note here that τ_1 is really much larger (worse quality) than τ, but only by a constant factor. □

A constant-bounded worsening in quality after a single insertion is intuitive and expected, but clearly insufficient to prove a global quality bound over the life of the algorithm. The single insertion bound will be used for moves that sacrifice quality to work on conforming. In between these moves are entire sequences of moves done to maintain quality, and much stronger inductive results hold for these cleaning sequences.

Lemma 19 (Cleaning Preserves Current Feature Size)**.** *Consider a mesh M', and* `SVRC` *dequeueing a series of insertions consisting only of* Skinny *work events to obtain some M'', then for every vertex $v \in M''$, it follows there is some constant Φ such that:*

$$f_{M'}(v) \leq \Phi n_{M''}(v)$$

Proof. The argument is inductive and similar to the Weak Spacing Theorem 5.

First claim that at the time of insertion of any vertex v,

$$f_{M'}(v) \leq (\Phi - 1) \underset{M''_v}{\mathrm{NN}}(v)$$

This fact has the obvious corollary that for any mesh M'',

$$f_{M'}(v) \leq \Phi \underset{M''}{\mathrm{NN}}(v)$$

(this is identical to the structure of Corollary 2 in the Weak Spacing Theorem.)

Suppose `SVRC` seeks to Destroy some skinny Voronoi cell. There are two cases, either `SVRC` insert a circumcenter c or warps to some input point u.

Suppose first that c is inserted, this is identical to Case 1 of the Weak Spacing Theorem.

$$f_{M'}(c) \leq (1 + \frac{2\Phi}{\tau}) n_{M''}(c)$$

So this requires $\Phi - 1 > (1 + \frac{2\Phi}{\tau})$ which will be strictly dwarfed by the second case.

The second case is when `SVRC` warps to some input point u.
This case differs from Weak Spacing, in that the insertion is charged to the proximity of the

small feature, not recursively to a lower dimensional mesh. Suppose $V(p)$ is being destroyed because it is skinny:

$$\begin{aligned} f_{M'}(u) \leq |up| + f_{M'}(p) &\leq (1+\theta)R_p + \Phi n_{M''}(p) \\ &\leq (1+\theta)R_p + 2\Phi r_p \\ &\leq (1+\theta+\frac{2\Phi}{\tau})R_p \leq (1+\theta+\frac{2\Phi}{\tau(1-\theta)}n_{M''}(u) \end{aligned}$$

Unravelling shows that setting $\Phi > (1+\theta)\tau/(\tau(1-\theta)-2)$ satisfies the lemma. Note here that this relies on the trivially satisfied $\tau \cdot (1-k) > 2$. □

Next is Lemma 20, which roughly states that the cleaning process is approximately monotone with respect to quality. While cleaning to improve the mesh quality from some τ_1 back to τ, `SVRC` will never reach an intermediate stage where the quality of the mesh is more than marginally worse.

Lemma 20 (Intermediate Quality)**.** *Suppose `SVRC` begins with a τ_1-quality mesh M and dequeues a series of* Skinny *work events, There exists a constant τ_2 depending only on τ_1, τ and θ such that every intermediate M'' is a τ_2-quality mesh.*

Proof. Consider M'', a τ_2-quality mesh obtained in the process of cleaning M'. First note as before that for every point x, $G_{M''}(x) \leq G_{M'}(x)$.
Note that by Lemma 19, it holds that for every vertex v,

$$f_{M''}(v) = n_{M''}(v) \geq 1/\Phi f_{M'}(v)$$

Since M' was τ_1-quality, combine with Lemma 8 to obtain that:

$$\Gamma_{M''}(v) \leq \Phi\Gamma_{M'}(v) \lesssim 1$$

Since $\Gamma_{M''}$ is bounded at every vertex, it then follows from Lemma 9 that M'' is a τ_2-quality mesh where $\tau_2 \lesssim 1$. □

Lemma 21 is the obvious corollary following from Lemmas 18 and 20.

Lemma 21 (Always Quality)**.** *At any point during `SVRC`, the intermediate mesh is a τ_2-quality mesh.*

Theorem 6 (Sparse Mesh)**.** *Any intermediate mesh during the lifetime of Sparse Voronoi Refinement is sparse, i.e. there is a constant depending only on τ and θ that bounds the degree of every vertex.*

Proof. This follows directly from the Degree Bound Theorem 2. □

4.1.3 Crease Recovery

Recall that SVRC adds balls to the system dynamically, with the goal of eventually covering the entirety of the creases. The goal of this subsection is to prove that this procedure in fact works, an essential part of the termination of the algorithm. An argument will show that the current mesh is not too small in the neighborhood of a crease before it is covered by the collar system. The notion of "not too small" in this lemma will also be useful later in proving that there are not too many extraneous vertices later removed from the collar region during RipStitchCollar, essential to the overall runtime of the algorithm.

Recall the following definition from the algorithm:

Definition 40 (Isolated Point)**.** *Consider a point $x \in \mathcal{P}$ and any mesh M. Let F be the parent feature of x, and v be the vertex of M such that $x \in V_M(v)$. The point x is* **isolated** *with respect to M and $\mathcal{P}$ if every feature of $\mathcal{P}$ that intersects $V_M(v)$ contains x.*

This definition is useful because it is easily testable during the run of SVRC. To relate this to the proof technology, the following simple lemma exposes a relation between isolation and the sizing g:

Lemma 22 (Isolated is Equivalent to g)**.** *Let x, $\mathcal{P}$, M, and v as in Definition 40. If $R_v \leq g(x)/2$ iff x is isolated.*

Proof. Follows directly from the definition of g. Clearly $B(x, g(x))$ does not intersect any features except those containing x. Consider $x \in B(v, R_v) \in B(v, g(x)/2)$, but then $B(v, g(x)/2)$ must be a subset of $B(x, g(x))$, and so x is isolated. Prove the other direction by contrapositve. Let w be a witness to $g(x)$, i.e. $g(x) = |xw|$. If x is not isolated, then WLOG $w \in V(v)$, so $g(x) = |xw| \leq |xv| + |vw| \leq 2R_v$. □

Now I prove the main lemma:

Lemma 23 (Crease Recovery)**.** *Every vertex v created on a crease of $\mathcal{P}$ is marshalled to the function* CreateCollar*. Furthermore, there exists some constant $\varepsilon_0 > 0$ such that the inequality $n_M(v) > \varepsilon_0 g(v)$ holds for the mesh M when* CreateCollar *is called.*

Proof. Let v be given. Recall from the algorithm that v is marshalled to CreateCollar if ever it is the case that v is θ_0-medial and is isolated, or if v is inserted.
Recall further that $\theta_0 := \frac{2\theta}{2\theta+1} \in (0, 1)$, where SVRC insertions warp to points within θ times the radius when inserting a circumcenter.

Take

$$\varepsilon_1 := \frac{1}{4\tau_2}$$

, where τ_2 is the global bound on mesh quality throughout the algorithm from Lemma 21.

Let x be the first vertex that is inserted into any mesh M such that $|xv| < \varepsilon_1 g(v)$. Consider the mesh M^- and M^+ immediately before and after the insertion of x.

If $x = v$, then the collar size is called immediately before v is inserted, and since v must have been $(1-\theta)$-medial (Lemma 18). Then by choice of x, it must be that $n_{M^-}(v) > \varepsilon_1 g(v) > \varepsilon_0 g(v)$, so the case is trivial. Assume $x \neq v$ for the remainder of the proof.

Clearly x must have higher containment dimension than v for it to be within $B(v, \varepsilon_1 g(v))$ and must be on some superfeature of v's parent feature. Thus, when x was created as the center of some gap-ball $B(x, 2r_x)$, it had a chance to warp to v, but did not. Thus

$$|xv| > 2\theta r_x \tag{4.1.4}$$

but then

$$\theta 2r_x < \varepsilon_1 g(v) \tag{4.1.5}$$

By Lemma 21, $R_x < \tau_2 r_x$, so then $R_x < 2\tau_2 \varepsilon_1 g(v)$. But then since $v \in B(x, R_x)$ by design, the choice of VE_1 gives isolation in M^+. by Lemma 22.

Since v is isolated in M^+, I now claim that v is θ_0-medial wrt to M^+.

Since x was the center of a gap-ball, there was some vertex b of M^- on the surface of the ball. Then by (4.1.4):

$$|bv| \leq |xv| + r_x \leq (1 + \frac{1}{2\theta})|xv|$$

Since x and b are both in M^+, we have $n_{M^+}(v) = |vx|$ and $f_{M^+}(v) \leq |bv|$, but then this yields:

$$n(v) \geq \frac{2\theta}{2\theta + 1} f(v) = \theta_0 f(v)$$

So v is θ_0-medial.

Thus CreateCollar will be invoked after the insertion of x (if not already at some previous iteration). It remains to show the lower bound on $n(v)$ at this iteration.

Take b as before, and take:

$$\varepsilon_0 := \frac{\theta \varepsilon_1}{\theta + 1}$$

Note since $\theta \in (0, 1)$, then $0 > \varepsilon_0 > \varepsilon_1$. Since x was the first vertex closer than $\varepsilon_1 g(v)$, then $|bv|$ must have been larger than this distance. Consider then, using (4.1.4) as well:

$$\varepsilon_1 g(v) < |bv| \leq |xv| + |xb| = |xv| + 2r_x < (1 + \frac{1}{\theta})|xv|$$

Bringing the constant over, then:

$$n_{M^+}(v) = |xv| > \frac{\theta \varepsilon_1}{\theta + 1} g(v) = \varepsilon_0 g(v)$$

Over the life of the algorithm, the nearest neighbor distance n only decreases. So if CreateCollar was called at some earlier iteration before the insertion of x, then the lower bound on n still holds. □

The previous lemma guaranteed that the vertices on creases are resolved early enough. A technicality is to make sure that the rest of the crease (along segments in particular) is protected as well. Recall the ε-collar-region from Section 1.4 as the region given by the union of balls $B(x, \varepsilon g(x))$ taken over all x on the creases. I will show this region is protected for some constant ε_2.

Lemma 24 (Crease Protection)**.** *There exists a constant ε_2, such that no* Unresolved *work event is ever enqueued in the ε_2-collar-region for a vertex not on a crease.*

Proof. Proof is by contradiction, and will rely on the lower-bound on size at crease recovery from the Crease Recovery Lemma 23. Take ε_0 as in the proof of Lemma 23 Take $\varepsilon_2 := \varepsilon_0/3 < \varepsilon_0/(2+\varepsilon_0) < \varepsilon_0$.

Suppose some vertex v has an Unresolved work event enqueued and v is in the ε_2-collar-region but not on a crease.

Suppose v near some created vertex on crease u, i.e. $v \in B(u, \varepsilon_2 g(u))$, but since $\varepsilon_2 < \varepsilon_0$, then by the Crease Recovery Lemma, CreateCollar has been called on u. Then since $v \in B(u, g(u)/3)$, then v is interior to the collar system and would not have been enqueued.

So it must be the case that v is near some crease subsegment but far from its vertices. Based on its relative distance to the segment and the definition of g, it is clear that v must be on some superfeature of the segment. But then v would not be inserted if it encroached on this segment. Suppose a is an endvertex of this segment, let R and c be the center radius of the diametral protective ball. Without loss of generality, assume that v is close to c, so that $|vc| < \varepsilon_2 g(v)$. Since v does not encroach, and g is Lipschitz along the segment, then:

$$R < |vc| < \varepsilon_2 g(c) \leq \varepsilon_2 g(a) + \varepsilon_2 R$$

Since the other end of the segment is not too close to a by Lemma 23, it holds that $\varepsilon_0 g(a) < 2R$, so this gives:

$$R < \varepsilon_2 g(a) + \varepsilon_2 R < \frac{2\varepsilon_2}{\varepsilon_0} R + \varepsilon_2 R < \left(\frac{2\varepsilon_2}{\varepsilon_0} + \varepsilon_2 \right) R$$

But then the choice of ε_2 makes the right side constant less than one, so this is a contradiction. □

4.1.4 Exterior Spacing

In this section, I will combine the weak spacing theorem along with the Crease Recovery Lemma to prove the Strong Exterior Spacing theorem, which will guarantee termination for `SVRC`.

Consider the ε_2-collar-region as in the previous, Section 4.1.3, and denote this region $\mathcal{I}$. Consider the **exterior vertices** outside of the collar system in the region $\mathcal{O} = \Omega - \mathcal{I}$. $\bar{g}$ is bounded away from zero in this region, and vertex spacing is bounded from below by $\bar{g}$ by the Weak Spacing Theorem 5 , so `SVRC` can only generate finitely many vertices in $\mathcal{O}$.

Let $\hat{M}$ be the (possibly infinite) set of points generated by `SVRC`.

Definition 41 (Exterior Vertices)**.** *Given any point set M, define the exterior vertices $\dot{M}$ as those whose Voronoi cells $V_M(v)$ intersect the exterior $\Omega - \mathcal{I}$. At any iteration i, define the exterior vertices as $M^i \cap \mathcal{O}$.*

The claim is that the actual spacing of `SVRC` is given by $f_{\hat{\dot{M}}}$, that is to say, all the spacing is governed by exterior vertices. For this section let τ_2 as in the Always Quality Lemma 21.

Lemma 25 (Exterior Size)**.** *At any time, for any points in the exterior, removing the interior vertices does not significantly change their feature size. Consider any M and let $x \in \mathcal{O}$, then:*

$$f_M(x) \leq \frac{\tau_2}{2+\tau_2} f_{\dot{M}}(x)$$

Proof. Let $v \in \dot{M}$ such that $x \in V_M(v)$, and let $w \in \dot{M}$ such that v, w are neighbors in M, note this implies that $|v - w| \leq 2R_v^M$.
Since $x \in V_M(v)$, then $|x - v| \leq f_M(x)$ and $r_v^M \leq f_M(x)$.
Using triangle inequality, τ_2-well-spaced, and these three facts, it follows:

$$f_{\dot{M}}(x) \leq |xw| \leq |xv| + |vw| \leq f_M(x) + 2R_v^M \leq f_M(x) + \frac{2}{\tau_2} r_v^M \leq (1 + \frac{2}{\tau_2}) f_M(x)$$

Moving the constant to the other side yields the lemma. □

By the previous lemma and by since subset nesting increases f, consider the following corollary:

Corollary 3 ($f_{\dot{M}}$ nested)**.** *For $x \in \mathcal{O}$, for any $M \subset \hat{M}$:*

$$f_{\dot{M}}(x) \geq f_M(x) \geq f_{\hat{M}}(x) \geq \frac{\tau_2}{\tau_2 + 2} f_{\hat{\dot{M}}}(x)$$

Inside the collar region for $y \in \mathcal{I}$, the final inequality does not hold, merely:

$$f_{\dot{M}}(y) \geq f_M(y) \geq f_{\hat{M}}(y)$$

I will lower bound the nearest neighbor on input by the function $f_{\hat{\dot{M}}}$.

Lemma 26 (Exterior Spacing)**.** *Suppose `SVRC` adds a new vertex u to some mesh M, then:*

$$f_{\hat{\dot{M}}}(u) \lesssim n_M(u)$$

Proof. The proof of this theorem is identical to the proof of the Weak-Spacing Theorem 5, with only the base cases changed. The lower bound side of all the inequalities will be $f_{\hat{\hat{M}}}$ instead of $\bar{g}$.

All of the inductive cases in the proof of Theorem 5 relied only on the Lipschitz condition for $\bar{g}$, which also holds for $f_{\hat{\hat{M}}}$, and so they are identical and will not be repeated here.

There were four base cases in the Weak-Spacing Proof. Case 2(A)(ii), when the nearest neighbor after an UNRESOLVED event was on a different feature. Case 3A, when a protective ball was encroached by an insertion on a superfeature. Case 4A, when a simplex on the collar surface was refined for bad representation angle or topology. And lastly, Case 4B, when a simplex with all but one vertices on the collar surface was refined for a COLLAR event.

In the Weak-Spacing proof, all four of these cases actually use g to establish a lower bound, and then use $\bar{g} \leq g$ to finish.

I then claim that if it is established that $f_{\hat{\hat{M}}} \lesssim g$ in all these insertion settings, then the Exterior Spacing Lemma is finished. Alternatively it may established directly that $f \lesssim n$ in some cases.

Suppose for the remainder of the proof that some vertex u is being inserted into some mesh M.

Case 2(A)(ii). Since this is an UNRESOLVED event, then by the Crease Protection Lemma 24, either u is on a crease or $u \in \hat{\hat{M}}$. If $u \in \hat{\hat{M}}$, then the lemma is trivially satisfied, since $f_{\hat{\hat{M}}}(u)$ will be the distance to u's final nearest neighbor which will be less than $n_M(u)$. Suppose u is on a crease. Then its distance to the exterior is lower bounded, i.e. $|u\mathcal{O}| \geq \varepsilon_2 g(u)$. But $f_{\hat{\hat{M}}}(u)$ is at least $|u\mathcal{O}|$, so the case is finished.

Case 3A, wherein one feature encroaches another is similar to the previous. If $u \in \hat{\hat{M}} \subset \mathcal{O}$ then we are done. Suppose then that $u \in \mathcal{I}$. Since u is the center of the circumball of the simplex, if any simplex vertex v is in $\mathcal{O}$ (and hence $\hat{\hat{M}}$), then $f_{\hat{\hat{M}}}(u) \geq |uv| = n(u)$. If all the vertices of the simplex are in $\mathcal{I}$, then u must be on a crease, but then $f_{\hat{\hat{M}}}(u) \geq |u\mathcal{O}| \geq \varepsilon_2 g(u)$ as before.

Case 4A and Case 4B both consider insertions on the collar surface. In this case these vertices are in $\mathcal{O}$ and hence in $\hat{\hat{M}}$, so once again $n_M \geq f_{\hat{\hat{M}}}$.

□

Theorem 7 (Termination)**.** *SVRC terminates with a mesh M.*

Proof. First, I claim the set of vertices $\hat{\hat{M}}$ is finite. By the Weak Spacing Theorem 5, the distance between these vertices is lower-bounded by $\bar{g}$. All of these vertices are in the exterior $\mathcal{O}$, and $\bar{g}$ is bounded away from zero everywhere in $\mathcal{O}$, so by compactness, $\hat{\hat{M}}$ must be finite.

It then follows that $f_{\hat{\hat{M}}}$ is bounded away from zero everywhere. By the Exterior Spacing Lemma 26, the spacing everywhere is lower-bounded by $f_{\hat{\hat{M}}}$, and since this bounded away

from zero, it then follows by compactness that SVRC terminates. □

4.1.5 Spacing Between Meshes

Consider now a spacing theorem that relates the relative spacing in different feature meshes at any fixed time iteration. The lemma shows that the spacing will always be similar among all the meshes.

Lemma 27. *Suppose p is considered for insertion into M_F at some iteration. Let $F' \supset F$ with $dim(F') = dim(F) + 1$. Then for some constant H:*

$$f_{M_F}(p) > H f_{M_{F'}}(p)$$

Proof. Write $M = M_F^i$ and $M' = M_{F'}^i$. Suppose in the first case that p was the center of a gap-ball $B(p, r)$ that was encroached by q. This is a base case. It follows from q not warping that:

$$n_{M'}(q) \leq \frac{2}{\theta} r = \frac{2}{\theta} f_M(p)$$

Point q was considered for insertion in M', so q was $(1-\theta)$-medial, i.e. $n_{M'}(q) \geq (1-\theta) f_{M'}(q)$. Since p was the center of gap-ball b, then $f_M(p) = r$. Using Lipschitz, combine to obtain:

$$f_{M'}(p) \leq f_{M'}(q) + |pq| \leq \frac{1}{1-\theta} n_{M'}(q) + r \leq (\frac{2}{\theta(1-\theta)} + 1) r = (\frac{2}{\theta(1-\theta)} + 1) f_{M_F}(p)$$

which will be satisfied if H large enough.

Suppose in the second case that p is a Voronoi corner of a skinny Voronoi cell belonging to q.
Apply the hypothesis inductively on q or its nearest neighbor of M^i, and obtain that $(1 + H) f_M(q) \geq f_{M'}(q)$ as in Corollary 2. Then:

$$f_M(p) = |pq| = R_q \geq \tau r_q = \frac{\tau}{2} f_M(q) \geq \frac{\tau}{2(1+H)} f_{M'}(q)$$

Then:

$$\begin{aligned} H f_M(p) &\geq \frac{\tau}{2} f_{M'}(q) - f_M(p) \\ &\geq \frac{\tau}{2}(f_{M'}(p) - |pq|) - f_M(p) \\ &= \frac{\tau}{2} f_{M'}(p) - (1 + \frac{\tau}{2}) f_M(p) \end{aligned}$$

Thus:

$$\frac{2}{\tau}(H + 1 + \frac{\tau}{2}) f_M(p) \geq f_{M'}(p)$$

Or simply:

$$(\frac{2H}{\tau} + \frac{2}{\tau} + 1) f_M(p) \geq f_{M'}(p)$$

□

So the lemma requires an H such that $H \geq \frac{2H}{\tau} + \frac{2}{\tau} + 1$, but this simply requires that $\boxed{H \geq (\tau+2)/(\tau-2)}$ Clearly, large enough H can be taken since $\tau > 2$.

Combining Lemma 27 over all dimensions, then there exist a constant (H^2) so that $f_{M_F}(p) \gtrsim f_{M_\Omega}(p))$ for any feature mesh M_F and any point $p \in M_\Omega$. In particular, lower bounds the size of any protective balls which are queued.

Corollary 4. *For any created protective ball $B := Ball(c, r)$ protecting a lower dimensional mesh M_F,*

$$r \gtrsim f_{M_\Omega}(c)$$

Proof. Since B is a created ball, it has two vertices of M_F on it, so $r = f_{M_F}(c)$. Then $f_{M_F} \gtrsim f_{M_\Omega}$ from the previous. □

4.2 Scaffolding

Frequently in meshing, and in SVRC, a region is meshed that is larger than the actually feature F being meshed. This is true for the bounding-boxes around features in SVRC and in many other algorithms. The extra vertices only exist to fill out space with a quality mesh, all of the mesh resolution is really due to F. In this setting, call the uninteresting space filling vertices **scaffolding**, they exist to support F, and will eventually be torn down.

The remarkable result of this section is that in most settings, the number of the scaffold vertices is only linear in the number of interesting vertices. SVRC will apply this in two ways. First, to guarantee that no extra work went into filling out the bounding-boxes Ω_F around every subfeature, since this is all discarded at the end. SVRC also discards any extraneous vertices that were added interior to the collar system. These are really just scaffolding for the exterior vertices, and are similarly bounded in size.

The scaffolding setting is general enough to be applied retroactively to almost every meshing algorithm that uses a bounding-box [BEG94, ABE98, She98, CDE+00, ELM+00a, MV00, MPW02b, Üng04, HMP06, CDR07]. Scaffolding also has a nice implication for surface meshing. Recall that SVRC is generating tetrahedralizations that fill space, but suppose a user only requires good triangulations of the features of the PLC $\mathcal{P}$. A user can run SVRC and discard all the simplices not on $\mathcal{P}$. This will have a smaller output-size, but scaffolding guarantees that SVRC still ran with output-sensitive efficiency. The results in this section are general dimensional with constants depending on dimension.

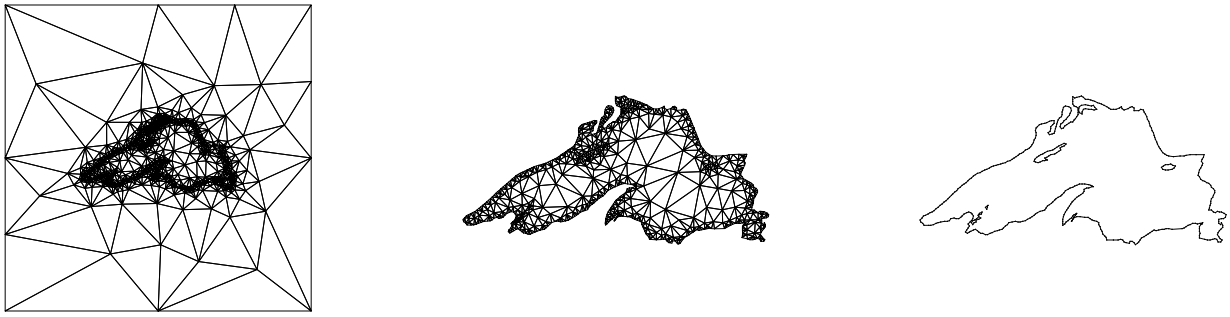

Figure 4.1: Incremental mesh refinement algorithms first generate a mesh over a bounding box (left), then remove the scaffold vertices and elements (center). Some applications are interested in only the surface mesh (right). The one-dimensional Lake Superior surface mesh shown has 530 surface vertices. The volume mesh shown has 1072 total volume vertices; 258 interior and 284 exterior. We offer the first theoretical analysis of the costs of this scaffolding.

The scaffolding result is reminiscent of one by Moore [Moo95]. A *balanced* quadtree is one in which neighboring quadtree cells have size within a constant factor of each other. Given an arbitrary quadtree with l leaves, Moore proved that the quadtree can be balanced by splitting only $O(l)$ cells. The operation of balancing a quadtree is analagous to filling space with a quality mesh.

For the rest of this section, consider a quality mesh M called a **scaffold mesh**. It has a subset N that is called a **driver set**. These will be defined formally, but the intuition is that all the spacing in M is only driven by spacing between the vertices of N.

4.2.1 Scaffolding Definitions

The main result is the Theorem 8, the Scaffold-Sizing theorem showing that given a scaffold mesh M with underlying driver set N, $|M|$ is bounded above by a constant times the size of $|N|$. Define now a formal setting:

Definition 42 (Scaffold Mesh, Driver Set)**.** *Suppose $M \subset \Omega$ is a finite vertex-set. Consider a* **driver set** *$N \subset M$. For $\tau \geq 1$ and $\alpha \in (0, 1)$, say that M is an (α, τ)-***Scaffold Mesh** *for N in Ω if two conditions hold. First, M must be τ-well-spaced in Ω. Second, N must drive all the mesh sizing thusly:*

$$\forall\ m \in M,\ \ f_M(m) \geq \alpha f_N(m) \tag{4.2.1}$$

Scaffolding requires that the domain Ω being filled is somewhat well-proportioned to the underlying driver set, otherwise filling out Ω could take arbitrarily many vertices. Enforce this by defining the notion of a **ρ-seed** of an embedded graph.

Definition 43 (ρ-seed)**.** *Suppose G is a graph embedded in Ω with vertices N. A* **ρ-seed** *N_0*

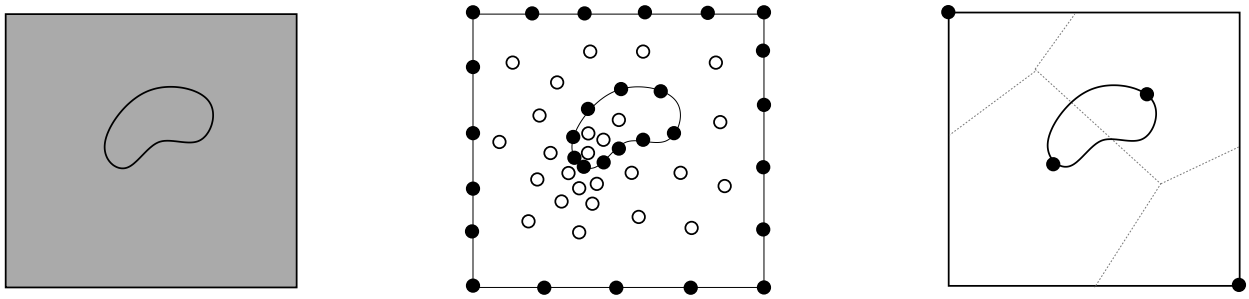

Figure 4.2: The definitions of Section 4.2.1 illustrated abstractly. **Left,** a surface $\mathcal{S}$ is composed of *both* black shapes, with the domain Ω shaded. **Center,** vertices form a scaffold mesh M of Ω. The subset of surface vertices $N_{\mathcal{S}}$ are shown in black, with the volume vertices in white. Observe the density of the volume vertices is driven only by the spacing of surfaces vertices. **Right,** a possible seed N_0, containing at least two points from each component. Notice the four points are ρ-well-spaced and have quality Voronoi cells (shown in dashed lines).

is a ρ-well-spaced subset of N containing at least two vertices for each connected component of G.

Definition 44 (Delaunay Sub-Diagram)**.** *Given point sets $N \subset M$, define the* **Delaunay sub-diagram** *of M restricted to N as* $\mathrm{Del}(M|N) := \mathrm{Del}(M) \cap \mathrm{Del}(N)$.

Note that $\mathrm{Del}(M|N) \neq \mathrm{Del}(N)$ in general.

4.2.2 Scaffold-Sizing Theorem

Theorem 8 (Scaffold-Sizing Theorem)**.** *Suppose M is an (α, τ)-Scaffold Mesh for N in Ω, and suppose* $\mathrm{Del}(M|N)$ *has a ρ-seed N_0, then:*

$$|M| \lesssim |N|$$

with constant only depending on α, τ, ρ, and dimension d.

Proof. N admits a well-paced ordering by Lemma 28 (to be proved), thus by Well-Pacing Theorem 3:

$$\int_{\Omega} \frac{1}{f_N^d} \lesssim |N| \tag{4.2.2}$$

Since M is τ-well-spaced, by Lemma 11:

$$|M| \lesssim \int_{\Omega} \frac{1}{f_M^d} \tag{4.2.3}$$

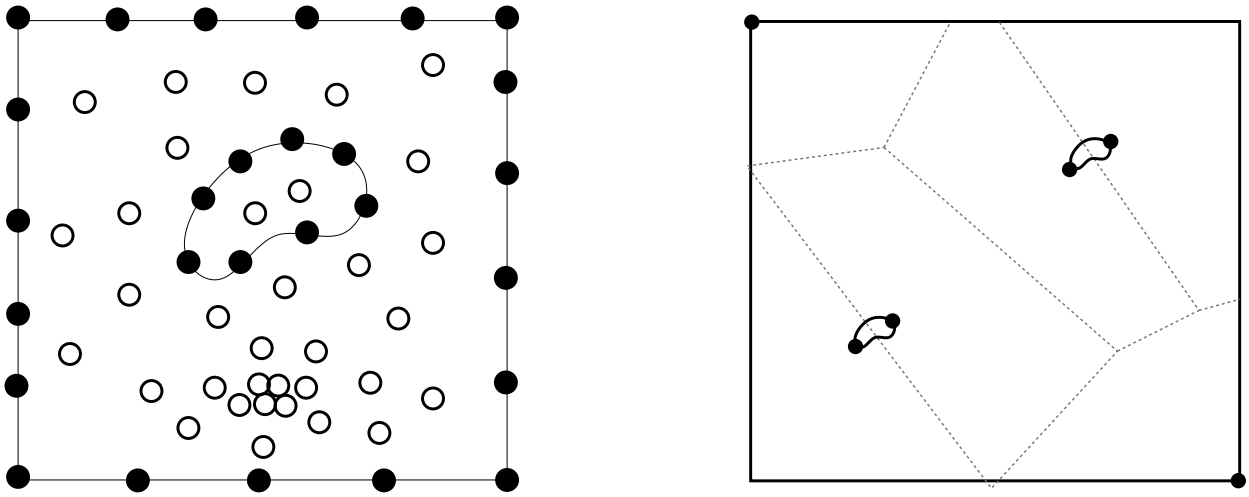

Figure 4.3: Examples of Non-Scaffold Meshes. **Left,** this volume mesh is not a scaffold mesh, because the sizing is not driven by the surface. The sink in the lower-center could contain arbitrarily many volume vertices. Note how this violates equation (4.2.1). **Center,** when the surface $\mathcal{S}$ has disproportionately small components, it will be too costly to fill Ω in a way that resolves these small surface features. Note that no seed can exist in this example. An attempted seed is shown, but as the surface components grow relatively small, there is no way to fit two points on each component in a way that is well-spaced.

But M is an scaffold mesh for N, so $f_M(m) \geq \alpha f_N(m)$ for $m \in M$. Consider any $x \in \Omega$, and let m be a nearest-neighbor of x, then by Lipschitz and Lemma 3:

$$f_N(x) \leq |xm| + f_N(m) \leq |xm| + \alpha f_M(m) \leq (1+\alpha)|xm| + \alpha f_M(x) \leq (1+2\alpha) f_M(x)$$

So $f_N \leq (1+2\alpha) f_M$ hence:

$$\int_\Omega \frac{1}{f_M^d} \lesssim \int_\Omega \frac{1}{f_N^d} \tag{4.2.4}$$

Combining (4.2.2), (4.2.3), and (4.2.4) gives $|M| \lesssim |N|$. □

Most of the work is in the remaining lemma:

Lemma 28 (N admits a well-paced ordering)**.** *Suppose M is a τ-well-spaced set in Ω, with a subset N such that* $\mathrm{Del}(M|N)$ *has a ρ-seed N_0. Then, N admits a (ρ, θ)-well-paced ordering with N_0 as the base set and $\theta = \frac{1}{2+3\tau}$.*

Proof. Construct an ordering by beginning with N_0, then selecting θ-medial vertices greedily, and prove by contradiction that there is always a θ-medial vertex that can be added to the current set.

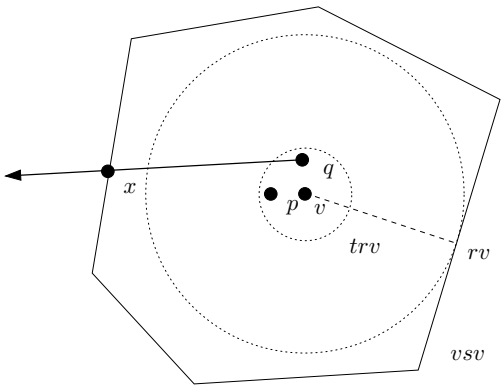

Figure 4.4: Figure for the proof of Lemma 28. If there is still a non-medial uninserted point p, then the original Voronoi cells from M were small near q but large near x, contradicting the assumption that M is well-spaced.

Suppose for contradiction that some vertex-set N_i is reached with $N_0 \subset N_i \subsetneq N$, and there no more unadded points are θ-medial with respect to N_i.
Let $p \in N - N_i$ and take $v \in N_i$ such that $p \in V_{N_i}(v)$.
p is not θ-medial, or it would be added, so it must be relatively close to v in the following sense. Using Lipschitz and Lemma 3

$$|pv| < \theta f_{N_i}(p) \leq \theta(|pv| + f_{N_i}(v)) = \theta|pv| + \theta r_v^{N_i}$$

Unravel this as:

$$|pv| < \frac{\theta}{1-\theta} r_v^{N_i} \tag{4.2.5}$$

Let e be an edge of $\mathrm{Del}(M|N)$ with one endpoint q in $V_{N_i}(v)$ and the other end outside $V_{N_i}(v)$, so that e exits $V_{N_i}(v)$ at some point x. Such an edge must exist, otherwise an entire connected component of $\mathrm{Del}(M|N)$ would be contained with in $V_{N_i}(v)$, which would be a contradiction since N_0 is a seed of $\mathrm{Del}(M|N)$.

It may be the case that q is equal to p, v, or neither of the two. Consider two cases with only slightly differing arguments, depending on whether $q = v$.

Case 1: Suppose $q = v$. Because x is on both the boundary of $V_{N_i}(v)$ and a Delaunay edge in M out of v, then $r_v^{N_i} \leq |xv| \leq 2R_v^M$. Using this along with Eq.4.2.5, it follows:

$$r_v^M \leq \frac{1}{2} n_M(v) \leq \frac{1}{2}|pv| \tag{4.2.6}$$

$$< \frac{\theta}{1-\theta} r_v^{N_i} \leq \frac{\theta}{1-\theta}|xv| \tag{4.2.7}$$

$$\leq \frac{2\theta}{1-\theta} R_v^M \leq \frac{1}{\tau} R_v^M \tag{4.2.8}$$

But this contradicts the assumption that M was τ-well-spaced.

Case 2: Suppose $q \neq v$. The second case is virtually the same except for one more degree of indirection. Then $q \notin N_i$ since $q \in V_{N_i}(v)$, so it follows (as before):

$$|qv| < \frac{2\theta}{1-\theta} r_v^{N_i} \tag{4.2.9}$$

The triangle inequality yields $r_v^{N_i} \leq |xv| \leq |xq| + |qv|$. Substituting this into Eq. 4.2.9, it follows:

$$|qv| < \frac{2\theta}{1-3\theta} |xq| \tag{4.2.10}$$

As before, since x on a Delaunay edge of M out of q, then $|xq| \leq 2R_M(q)$. Using this and Eq. 4.2.10, so:

$$r_q^M \leq \frac{1}{2} n_M(q) \leq \frac{1}{2} |qv| \tag{4.2.11}$$

$$< \frac{\theta}{1-3\theta} |xq| \leq \frac{2\theta}{1-3\theta} R_q^M \tag{4.2.12}$$

$$= \frac{1}{\tau} R_q^M \tag{4.2.13}$$

But again this contradicts the assumption that M was τ-well-spaced. Thus, there is always a θ-medial point to add, so N admits a well-paced ordering. □

Chapter 5

Runtime Analysis of SVRC

For an input PLC $\mathcal{P}$ with N features, spread Δ, and an output mesh M, the overall runtime of SVRC will be given by $O(N \log \Delta + |M|)$. There are two essential types of operations in SVRC, those that are linear with respect to the output size, and those that carry a $\log \Delta$ term.

All of the log term operations will be those related to point location, and are charged according to the same amortized argument. Every point location operation will be packed around a vertex on some feature, and these events will be charged to the feature, so that there will only be $\log \Delta$ operations per feature. If SVRC were just handling point-set input, this would be sufficient, but for PLCs there is more subtlety. Some operations will be clustered around a vertex that was *created* on a feature. If the total number of vertices created on features is denoted M_1, so that $N < M_1 < |M|$, then the simple amortized argument would only give a runtime bound of $O(M_1 \log \Delta)$ for point-location type operations. This is circumvented by showing that for vertices in $M_1 - M$, the number of operations packed around such vertices is bounded by a constant, instead of just $O(\log \Delta)$. The packing arguments used in this chapter will all build on Section 2.1.2.

5.1 Linear Work Operations

This section will be concerned with those operations of SVRC which will be linear in the size of the final output mesh M. Recall that SVRC actually generates many more vertices than will be output. Each feature mesh M_F is wholly discarded, and vertices are also discarded from the interior of the collar system by RipStitchCollar.

Proceeding in two stages, first let M_0 be the total number of vertices generated over the life of the algorithm. First, I will show that the non-point-location work of SVRC is linear in M_0, this will primarily rely on the bounded degree in each mesh (Theorem 2.) Second, I will show that $M_0 \lesssim |M|$. The non-point-location work is that of updating the mesh data

structure, maintaining flags on uninserted vertices, maintenance of the work queues, and some of the collar sizing work.

5.1.1 Refinement Work

The work for updating the mesh data structure during refinement is located in the method ForceInsert (Figure 3.3).

Lemma 29. *The function* ForceInsert *has runtime bounded by a constant, excluding the work in subroutines.*

Proof. Suppose ForceInsert is inserting p into M^- to form M^+. Recall both M^- and M^+ have bounded degree via the Always Quality Lemma 21 and Degree Bound Theorem 2.

Any Voronoi cell from M^- that will change can only do so because it is adjacent to the p in M^+, but there are only constantly many. The new vertex p and then be inserted via the Bowyer-Watson Delaunay insertion strategy [Bow81], which incrementally digs out a cavity and replaces it with a star around the new vertex. Dual operations can be simultaneously made to a Voronoi data structure, and the whole process runs in time proportional to the new degree of p, which is constant.

The check to see if $p \in$ Creases can be done with a flag on every uninserted vertex. This can be easily done by checking the parent feature when p is originally created as a circumcenter. The check to see if each modified cell is τ-skinny can be done in runtime proportional to each of their degrees, all of which is constant. □

5.1.2 Work Queues

`SVRC` needs to perform work events in an approximately largest first fashion, i.e. When a work event seeks to destroy some gap-ball with radius r, it must be that case that there is some constant so that $r \gtrsim r'$ for every other enqueued work event. To this end, the work queues in `SVRC` are to be implemented as a bucketed priority queue, where work events are bucketed according to $\lfloor \log_2 r \rfloor$. The queues will be implemented as simple doubly-linked lists of buckets with a head pointer. The operations on each queue will be Enqueue, Dequeue, and DecreaseSize; and each of these operations will run in constant time. Naively, the simple list implementation would require too much time, but `SVRC` will only perform queue operations with a restricted set of parameters that will guarantee the desired runtime. (These results bucketed priority queues for meshing were developed from unpublished research done with Benoît Hudson).

Lemma 30 (Queue Runtime)**.** *Suppose a queue has as its head a work event of size R_0. Then the queue operations require the following amounts of work:*

- Dequeue *requires constant time*

- ENQUEUE($B(c, R_1)$) *requires constant time if* $R_1 > R_0$*, otherwise time in* $O(\log R_0/R_1)$
- DECREASESIZE($B(c, R_1)$) *for an event with previous size* R_3 *requires time in* $O(\log R_3/R_1)$

Proof. These all follow directly from the naive queue implementation and simply walking down the linked list of buckets to perform insertions and size changes. □

The bounds on changes in size will follow directly from Section 4.1.2.

Lemma 31 (Bounded Size Changes)**.** *Every* ENQUEUE *and* DECREASESIZE *operation runs in constant time.*

Proof. Consider a queue with head size R_0, and consider inserting a vertex v to go from mesh M^- to M^+. Then it must be that $f_{M^-}(v) \gtrsim R_0$. From the proof of Lemma 18 (Eq. (4.1.3)) and from Lemma 19, it must be that $f_{M^+} \gtrsim f_{M^-}$. So if any cell u is changes size, then R_u could have gone down by only a constant multiplicative factor, so the time for DECREASESIZE will be constant. If any cell u is newly enqueued, then u must be v or adjacent to v. In either case, the Always Quality Lemma 21 gives $R_u \gtrsim R_v \gtrsim R_0$, so the new work event must be near the top of the queue and will only require constant work. □

5.1.3 Total Vertices Created

Lastly, it remains to bound the total number of vertices created by a constant times the output size $|M|$. First, I will bound the number of extraneous vertices in the feature meshes and their bounding boxes. Then I will bound the vertices inside the collar system that are discarded by `SVRC`.

Lemma 32 (Feature Mesh Vertices)**.** *The total number of vertices in all feature meshes is bounded by a constant times* $|M|$*.*

Proof. Let M_F be a feature mesh. When `SVRC` terminates, there will be no more UNRESOLVED work events, so any vertex $v \in M_F \cap F$ will also appear in M.

There are other vertices in M_F, namely those in the bounding box, but these can be accounted by a scaffolding argument. Recall the definitions of Section 4.2. Consider the subset $I = M_F \cap F \subset M_F$ consisting of the internal vertices. Clearly I the Delaunay sub-diagram $\mathrm{Del}(M_F \mid I)$ is connected and has a ρ-seed (Simply take two points that are separated by distance proportional to the diameter of F). I claim that M_F is a scaffold mesh with driver set I.

Recall the Weak Spacing Theorem, i.e that $f_{M_F} \gtrsim \bar{g}$. Observe that for $x \notin F$, by definition $\bar{g}$ will be at least the distance to the feature, so that $\bar{g}(x) \geq |xF|$. But all the vertices of I are in F, so that $f_M(x) \geq |xF|$. Combining, this yields, for every $x \notin F$:

$$f_{M_F}(x) \gtrsim \bar{g}(x) \geq |xF| \geq f_I(x)$$

Consider then any point $y \in F$. Because M_F is τ-well-spaced, the removal of the exterior vertices (those not in I) will not significantly increase f at any points in the interior, i.e. $f_{M_F}(y) \gtrsim f_I(y)$. (This is straightforward and equivalent in nature to Lemma 25.)

Since it then holds that $f_{M_F} \gtrsim f_I$ everywhere, then I is a driver set for the scaffold mesh M_F, thus by Scaffold-Sizing Theorem 8, $|M_F| \lesssim |I|$. But since $I \subset M$, the lemma is proved. □

The other vertices to consider are those that are discarded from inside the collar system during RipStitchCollar. The argument is similar to the previous lemma, only now the interior of the collar regions is charged to all the exterior vertices.

Lemma 33. *Let M_c be the all the vertices including those in the collar region before some are discarded to obtain output mesh M, then $|M_c| \lesssim |M|$.*

Proof. This proof is straightforward from the sizing arguments that have already been established.

Let O be those vertices of M which are outside the collar region, these will all be maintained, so that $O \subset M$. The Exterior Spacing Lemma 26 establishes that $f_{M_c} \gtrsim f_O$.

This the outside vertices O form a driver set for the scaffold mesh M_c. The Delaunay sub-diagram of $\mathrm{Del}(M_c \mid O)$ will be connected and so the bounding box forms a trivial ρ-seed. Thus the Scaffold-Sizing Theorem applies, so that $M_c \lesssim O$. But $O \subset M$, so the lemma is finished. □

5.2 Collar Sizing Runtime

The calls to method FastCollarSize generate work to perform an exploration of the local neighborhood around some vertex v in order to calculate $g(v)$.

The efficiency of this procedure is what causes the need for approximately largest first work queues. When the queues are processed largest first, the geometric sizing of the mesh will decrease in somewhat of a breadth-first fashion. If there are two disjoint regions that both require refining, then the largest-first order will roughly alternately refine them, rather than refining one first and then the other. This will be captured in the proofs by observing that in this case, the outstanding Unresolved work events will always be proportional to each other, and all the other cells in the mesh will be larger than this. The following lemma encodes this notion:

Lemma 34 (Same Size Lemma)**.** *Consider the mesh M at some point during a run of `SVRC`. Let R be the outradius of the largest* Unresolved *work event. Then for every vertex $v \in M$, $R_v \gtrsim R$.*

Proof. First, I claim that every UNRESOLVED work event so far has had radius larger than R. Suppose v is the unresolved feature causing the current largest work event. Any previous cell containing v must have had a larger radius, and so whatever UNRESOLVED work events have been done must have been even larger.

Since there is an outstanding UNRESOLVED event, then no COLLAR moves have been performed, so then UNRESOLVED events represent the only base case for the Weak Spacing proof. Letting R' be the radius of the smallest unresolved event performed, a simple revisiting of the Weak Spacing Proof then gives $n_M \gtrsim R'$. Alwasy

But since $R' > R$, then $n_M \gtrsim R$. The proof is then finished since it is always the case that $R_v \gtrsim n_M(v)$. □

Returning to the collar exploration, the procedure explores a local patch of the mesh looking for a witness to the function g. All of the work looking at witnesses will be accounted for in the next section. It is necessary to account for all the cells explored, in case there are many cells but few witnesses.

Lemma 35 (Fast Collar Exploration)**.** *Suppose* FASTCOLLARSIZE *is called at some vertex v. Then the exploration to find $g(v)$ explores at most constantly many Voronoi cells of M.*

Proof. Recall that v may or may not have just been inserted. Let M be the current mesh, unless if v was just inserted, then let M be the mesh just prior to inserting v.

Let $q \in M$ such that $v \in V(q)$. Since $q \neq v$, then $V(q)$ must be enqueued as an UNRESOLVED move. Let R be the outradious of the largest UNRESOLVED move, then $R \geq R_q$. But by the Same Size Lemma, every cell $V(u)$ in M has $R_u \gtrsim R \geq R_q$.

By the Crease Recovery Lemma, there is an empty ball around v of radius $\varepsilon_0 g(v)$, but this then implies that $|vq| \geq \varepsilon_0 g(v)$, so then $R_q \geq VE_0 g(v)$.

But this implies that every cell in the mesh is large with respect to $g(v)$, and since the cells are all τ_2-well-spaced, then only constantly many cells will intersect the ball $B(v, g(v))$. The breadth-first search will only try cells intersecting this ball before it finds a witness to $g(v)$, so the lemma is proved. □

5.3 Amortized Work Operations

The operations in `SVRC` related to point-location will be accounted in this section. Recall that every call to DESTROY results in a mesh insertion near the center of some gap-ball. Before this vertex is inserted, `SVRC` makes a query to the Bipartite Location Graph 3.2.3 to determine if there are any nearby location objects (protective balls and collars), either for warping, or for yielding to first perform a lower dimensional insertion. After an insertion eventually occurs, the BLG must be updated to reflect the newly changed mesh. Additionally, the flags for isolation and θ-medial must be updated for any UNRESOLVED events. The BLG is

also queried when FastCollarSize is called to determine the sizing $g(v)$ for a new collar centered at v. Call v a **target** when either v is being inserted or v is the center of a collar sizing query.

All of the work will occur in a relatively local neighborhood of the target, and so I will account for all this work as a set of **location work events** (LW-Events), each of which is a pair $\langle v, O\rangle$, where v is a target, and O is a location object of the BLG. There may be more than one LW-Event for a given pair, but I claim that the number of repeated events is constant per pair.

Lemma 36 (Unique LW-Events)**.** *The total number LW-Events is at most a constant times the number of unique pairs $\langle v, O\rangle$ for which there exists an LW-Event.*

Proof. The lemma follows by simple inspection of the algorithm. LW-Events occur in ForceInsert, UpdateLocation, PropagateLocation, CreateCollar, and FastCollarSize. Each of these methods is only ever called once with a given target. Furthermore, any pair $\langle v, O\rangle$ will cause at most constantly many LW-Events in each method call. Thus, the total LW-Events for any given pair is constant, so it suffices to count the unique pairs. □

Unfortunately, the number of LW-Events associated with a single target can be very large. Consider the first few mesh insertions, which may scan the entire input to update the BLG. The key argument is an amortization to count LW-Events per object, instead of per target. Let W be the set of all unique LW-Events. Given W and a single location object O, define the set of all LW-Events associated with O as W_O; simply those pairs which include O.

Some location objects may cause many LW-Events. Consider an Unresolved work event for some input vertex that is not resolved until the mesh reaches very fine resolution. This vertex will be checked for relocation many times as the mesh insertions decrease the size of whatever current Voronoi cell contains the vertex. In this case, I will show that the size of $|W_O| \lesssim \log \Delta$ if O is part of the input. On the other hand, consider a RepBall around some subsegment that is created at some time during the algorithm. Either this ball is part of the output, or soon after, this ball will be encroached. If it is encroached, then its center is inserted, it is destroyed, and two new balls are created. Either way, the lifetime of this object will be short enough that I will bound $|W_O| \lesssim 1$ for any created objects. Putting these together will give the runtime bound desired:

$$\begin{aligned} |W| = \sum_O |W_O| &= \sum_{O \in \text{Input}} |W_O| + \sum_{O \in \text{Created}} |W_O| \\ &\lesssim \sum_{O \in \text{Input}} \log \Delta + \sum_{O \in \text{Created}} 1 \\ &\leq N \log \Delta + |M| \end{aligned}$$

5.3.1 Counting Location Work

For this section, recall the lemmas on Structural Properties of Quality Voronoi Diagrams from Section 2.2. Recall from Lemma 21 that every mesh during `SVRC` is τ_2-well-spaced for some τ_2. This section will separate LW-Events into two cases based on their target.

Consider first the LW-Events where the target is a vertex v being inserted. Vertex v was inserted because of some gap-ball $B := B(c, R)$. The location objects that v will pair with will be any object intersecting B, as well as any object intersecting a Voronoi cell that is modified by inserting v.

The first lemma states that the such LW-Event pairs are not relatively far apart:

Lemma 37. *Let v, B, c and R, as in the preceding paragraph. Let O be an object such that $\langle v, O\rangle$ is a LW-Event, then:*

$$|vO| \lesssim R$$

Proof. Let M^- and M^+ be the mesh immediately before and after the insertion of v Since c is a Voronoi corner of M^-, let q be a vertex whose corner it is, so that $R = R_q(M^-)$.

Suppose first that O intersects some cell $V(u)$ that happened to intersect the gap-ball B. Let U be the set of all such u. By the Bounded-Ply Lemma 6, since all these cells intersect a gap-ball, then $|U| \lesssim 1$. But then by the Neighborhood Corollary 1, $R_u \lesssim R_q = R$ and $|uq| \lesssim R$ for every u. But then using triangle inequality, and since O intersected $V(u)$:

$$|vO| \leq |vc| + |cq| + |qu| + |uO| \leq R + R + |qu| + R_u \lesssim R$$

Now suppose instead that O intersected some cell $V_{M^-}(u)$ which is then modified by inserting v. It must then that u and v are adjacent in M^+. Also, in the new mesh M^+, either O intersects $V_{M^+}(v)$ or $V_{M^+}(u)$. If it is the former, then using $r_v(M^+) \leq |vq|/2 \leq R$:

$$|vO| \leq R_v \leq \tau_2 r_v \leq \tau_2 R \lesssim R$$

If it is the latter, then using the Neighbor Lemma 10 in M^+:

$$|vO| \leq |vu| + |uO| \leq 2R_v + R_u \leq 2R_v + \tau_2 R_v = (2 + \tau_2)R_v \leq \tau_2(2 + \tau_2)R \lesssim R$$

□

Since they are close together, then packing arguments developed in Section 2.1.2 will apply. Recalling Definition 19 for a Proximal Event Sequence, where each event (in this case a target) has an empty ball around it lower bounded by the distance to the center (in this case the location object):

Lemma 38 (Insertion LW-Events form a sequence)**.** *Consider any location object O and consider the set W_O of all the LW-Events $\langle v, O\rangle$ for which the target v is a vertex being inserted. Then W_O forms a Proximal Event Sequence around the O.*

Proof. Let some $\langle v, O\rangle$ be given. Let M^- be the mesh immediately before inserting v, and let $B := B(c, R)$ be the gap-ball being destroyed by v. Then v must have been near c, i.e. $|vc| \leq \theta R$. But since B was a gap-ball, this means that $n_{M^-}(v) \geq (1-\theta)R$. Let $B_v := B(v, (1-\theta)R)$ be the associated empty ball centered at v.

This ball is empty of any previously inserted vertices, so the sequence of pairs $\langle B_v, v\rangle$ form an event sequence.

But then by the preceding Lemma 37, it follows:

$$|vO| \lesssim R \lesssim (1-\theta)R$$

So this must be a proximal event sequence around O. □

A similar pair of lemmas will establish the same result for LW-Events whose target is performing a collar sizing query.

Lemma 39. *Suppose $\langle v, O\rangle$ is an LW-Event where v is the argument to* FastCollarSize*, then $|vO| \lesssim g(v)$*

Proof. Recall from the code (Figure 3.3) that `SVRC` performs a breadth-first search around v until it finds a witness for $g(v)$.

Let q be the Voronoi cell containing v, i.e. $v \in V(q)$, noting that it may be that $v = q$ if v was just inserted. This code is only called when v is isolated, therefore $R_q \leq g(v)/2$ by Lemma 22.

Suppose some cell $V(u)$ is explored. If $u = q$, then simply $|vO| \leq |vq| + |qO| \leq 2R_q \leq g(v)$. Suppose $u \neq q$. Then it must be the case that $V(u) \cap B(v, g(v)) \neq \emptyset$ otherwise this would contradict breadth-first; `SVRC` would first have explored all the cells intersecting $B(v, g(v))$ and would have found a witness and stopped. Let $x \in V(u) \cap B(v, g(v))$, then since u is the nearest-neighbor of x, $V(u)$ cannot be too large, i.e.:

$$R_u \leq |ux| \leq |xq| \leq |xv| + |vq| \leq g(v) + R_q \leq 1.5g(v)$$

Continuing then:

$$|vO| \leq |vx| + |xu| + |uO| \leq g(v) + 2R_u \lesssim g(v)$$

□

Continuing as in the other target case:

Lemma 40 (Collar Sizing LW-Events form a sequence)**.** *Consider any location object O and consider the set W_O of all the LW-Events $\langle v, O\rangle$ for which the target v is a vertex argument to* FastCollarSize*. Then W_O forms a Proximal Event Sequence around the O.*

Proof. Let some $\langle v, O\rangle$ be given. By the Crease Recovery Lemma 23, there is an empty ball around v with radius at least $\varepsilon_0 g(v)$ for some fixed ε_0 when FastCollarSize is called.

These balls are disjoint from any previously inserted vertices, so the pairs $\langle B(v, \varepsilon_0 g(v)), v\langle$ form an event sequence. But then by the preceding Lemma 39, it immediately follows that they form a proximal event sequence around O. □

5.4 Counting the Proximal Event Sequences

The previous section established that the LW-Events form two proximal event sequences around the location objects, one for insertion targets and one for collar targets. From here on, it suffices to consider one proximal event sequence around each location object, with the understanding that the total number of LW-Events is then underestimated by at most a factor of two.

Lemma 41 (Bound on LW-Events)**.** *Consider a location object O, and let M_O be the LW-Events associated with O. If O is a created object, then $|M_O| \lesssim 1$. If O is an input object, then $|M_O| \lesssim \log \Delta$.*

Proof. Let v be the target of the LW-Event which minimizes $|vO|$, with the exception that if O is a protective ball and v encroaches it, this should not be included. This excludes at most two LW-Events per object, since the ball would always be destroyed after it is doubly-encroached. If the location object is a ball $B(c, R)$, then this guarantees that $|vc| > R$.

Suppose O is a created object, then by Corollary 4, it is always the case that O is not too small, i.e. $R \gtrsim f_M(c)$ for every M containing O, which then implies that $|vc| \lesssim R$. But then all the LW-Events are at distance proportional to R, so the second statement of the Proximal Packing Lemma 4, so $|W_O| \lesssim 1$.

Suppose O is an input object. Then the farthest LW-Event cannot be larger than the diameter of the PLC. The nearest event will be no closer than the nearest neighbor in the final output. By the Exterior Spacing Lemma 26 this is bigger than $f_{\hat{M}}$ which is bigger than the smallest feature of the clipped PLC. Thus the ratio for farthest to nearest LW-Events is at most a constant times Δ, so by the Proximal Packing Lemma, there are at most $|W_O| \lesssim \log \Delta$. □

Corollary 5 (Total Location Work)**.** *The total number of Location Work Events W is bounded by:*

$$|W| \lesssim N + |M| \log \Delta$$

Chapter 6

Conclusions and Extensions

Overall, the SVRC algorithm represents the first algorithm for meshing arbitrary PLCs in three dimensions with an efficient runtime guarantee. There are a few limitations of SVRC as it has been presented herein. The first is the quality guarantee on the output mesh. SVRC can terminate for any Voronoi aspect ratio $\tau > 4\sqrt{2}$ for elements away from the creases. This gives a bound of $2\sqrt{2}$ on the radius-edge ratio of any output simplex. The state of the art for guaranteed quality in three dimensions is a bound of 2 [CDRR04]. It is still a challenge to reach this quality bound with efficient runtime.

Another limitation of SVRC may be the expense of the predicates needed for testing and maintaing the restricted Delaunay of the collar surface, testing for topological ball property, and good representation angle. Although these predicates will run in constant work, they may be expensive to implement in practice. One solution is to replace the explicit testing with a simple size criteria relative on the radius of the collar ball. This will still guarantee termination, but it will certainly generate more vertices on the collar surface. This may be an acceptable penalty in the interest better runtime.

6.1 Other Approaches

6.1.1 Weighted Delaunay Refinement

The approach to handling the collar surface in SVRC to explicitly maintain appropriate topology and no large angles adjacent to the creases continues from the work in [CP06, PW05]. There is another approach to handling the collar region using **Weighted Delaunay Triangulations** [CDR07]. The weighted delaunay triangulation assigns a weight to every vertex and has as its dual the power diagram, where the region is subdivided according to a weighted distance from regions. This weighting allows some vertices and edges to be forced into the weighted Delaunay.

Weighted Delaunay allows the proper output mesh topology around the creases to be enforced implicitly, so that the mesh will properly conform without a need for the RipStitchCollar routine.

This approach could easily be grafted into `SVRC` without affecting the runtime. The routine FastCollarSize can be used to find the appropriate weight for a vertex on a crease instead of the sizing for a collar ball. The effect of protecting the crease to guarantee termination would be the same.

An advantage of this approach is that it would save a large constant factor with regards to all the calculations performed by `SVRC` with respect to the collar surface; representation angles and topological ball property would not be computed. The main disadvantage of this approach is that it gives no guarantee on the tetrahedra adjacent to the crease. In general they may have angles as large as $\pi - \alpha^*$, where α^* is the smallest angle of the input PLC, whereas `SVRC` allows the an arbitrary bound σ to be input.

6.1.2 Two Pass Algorithms

One way to simplify algorithm `SVRC` would be to run it in two passes. The first pass would run until the collar system had been resolved. Then a clipped PLC could be computed, and the generic `SVR`without handle creases could run on this new PLC, with a post-process to stitch in an appropriate mesh where the creases used to exist. This approach would generate asymptotically the same sized output mesh, but most likely fewer vertices in practice, since the slack in mesh size from dynamically computing the collar region is removed.

The disadvantage of this approach would be that the new clipped PLC would have a larger description complexity than the original input. In general, if `SVRC` would generate M_1 vertices on the creases, then the clipped PLC would have $\Theta(N + M_1)$ features. Thus, the second pass would run in $O((N + M_1) \log \Delta + M)$. This would be desirable if there were few creases and minimizing output mesh size was a high priority.

6.2 Extensions

6.2.1 Slivers

Historically, many algorithms have been proposed that take as input a good radius-edge 3D mesh, and refine it into a sliver-free, good aspect-ratio mesh [Che97, ELM$^+$00b, LT01].

The original strategy of [Che97] was that whenever taking the center of a gap-ball for insertion, try several randomly selected points very near the center in attempt to avoid creating any new slivers smaller than the gap-ball being destroyed. Then add a new type of work event (Sliver) with the lowest priority, that destroys any slivers by adding a new vertex near their circumcenter.

In [LT01], Li and Teng showed that this notion extends to handle features by requiring that whenever a lower-dimensional ball is split, the algorithm can chooose randomly in a disc near the circumcenter, where the disc is restricted to lie on the lower-dimensional feature being refined. To further extend this into `SVRC`, it must handle the case where a RepBall is refined. Recall that a RepBall is centered on the collar surface. The new point must be chosen randomly near the center, and restricted to lie on the collar surface as well.

Using the analysis techniques of this thesis, a very simple timing analysis of the Li-Teng algorithm shows that it will work, only adding linear time and space requirements.

6.2.2 Curved Surfaces

Algorithm `SVRC` is not far from handling the more general case of Piecewise Smooth Complexes. Indeed, the predicates developed for handling the collar surface (which was essentially a PSC), can be extended to handle a PSC as input.

The first idea is that each bounding-box mesh M_F would be a full three-dimensional mesh, even if the feature F is only a one or two dimensional surface. Then the restricted delaunay of M_F restricted to F would be maintained, and the representation balls would be reported to higher dimension as protective balls for conforming.

These three-dimensional bounding box meshes around each feature will generate many excess vertices that will be discarded. Fortunately, scaffolding arguments still hold regardless of the dimensions of the scaffold mesh relative to its driver set. So a one-dimensional curve meshed in three dimensions still generates only linearly many excess vertices.

6.3 Improving $\log \Delta$

A major area for improvement would be the reduction of the point location costs from $O(N \log \Delta)$ down to $O(N \log N)$ for inputs with pathological spread. For a comparison based model, a sorting lower bound for triangulation applies, so that any algorithm must run in $\Omega(N \log N)$. If the point-location costs could be improved, then this would give `SVRC` the optimal output-sensitve runtime. Although this is largely a theoretical concern, it may lead to practically faster algorithms.

The factor of $O(\log \Delta)$ arises as the number of times an input vertex may be relocated over the life of the algorithm. This was bounded by observing that every relocation can be charged to some mesh insertion that made progress on increasing the mesh resolution. The progress ends up being a multiplicative factor, and the final resolution was at most the spread of the input, and hence the $O(\log \Delta)$ relocations.

This notion of "progress" is geometric in nature. In order to drop the factor to $O(\log N)$, the algorithm must make some *combinatorial* progress every time a relocation occurs. Mesh

operations must provably be subdividing the input in a combinatorial rather than a geometric fashion.

One approach would be some sort of median finding. In the early phases of the algorithm, there is a great deal of freedom in choosing a vertex for warping. Choosing a median breaks the input in half in one dimension, and with some other techniques can lead to the desired $O(N \log N)$ relocation costs. In higher dimension, some appropriate notion of median is needed. Rather than explicitly finding some type of median, the problem may also be solved by randomization. Either case requirs improved notions of combinatorial depth for points in dimension three and higher, especially in the case of a non-laminar series of subdivisions as those produced by unstructured meshing.

Chapter 7

Glossary

Global Symbols:

Symbol	Names and Description
$n_{\mathcal{X}}(x)$, n	Definition 17, Nearest Neighbor Distance, $\lvert x\mathcal{X}\rvert$
$f_{\mathcal{X}}(x)$, f	Definition 13, Second Nearest Disjoint Neighbor of x from $\mathcal{X}$
$g_{\mathcal{X}}(x)$, g	Definition 14, Largest Ball at x hitting two features of $\mathcal{X}$, one of which doesn't contain x
$\bar{g}_{\mathcal{X}}(x)$, $\bar{g}$	Definition 39, Weak Spacing Function, Second Nearest Neighbor
Ω	Closed, Bounded, Convex Subset of $\mathbb{R}^3$
$\mathcal{P}$	Input Piecewise Linear Complex
τ	Quality Bound Away from Creases, Desired Voronoi Aspect Ratio of the Output
σ	Quality Bound At Creases, Desired Maximum Representation Angle, Definition 34
$G_M(x)$	Definition 21, Gap-Size, Largest Ball with x on the surface, that is disjoint from M
$\Gamma_M(x)$	Definition 22, Grading, $(G_M(x)/f_M(x))$
θ	`SVRC` warps to $(1-\theta)$-medial points, function of τ
θ_0	`SVRC` creates collars when centers are θ_0-medial, function of θ
$V_M(v)$, $V(v)$	Voronoi Cell centered at v in the Voronoi diagram
$\mathrm{Del}(M)$, $\mathrm{Vor}(M)$	Delaunay and Voronoi Diagram of points M
$R_v(M)$, R_v	Definition 12, Outradius of the cell $V_M(v)$
$r_v(M)$, r_v	Definition 12, Inradius of the cell $V_M(v)$

Chapter 8

Bibliography

Bibliography

[ABE98] Nina Amenta, Marshall Bern, and David Eppstein. The crust and the β-skeleton: Combinatorial curve reconstruction. *Graphical models and image processing: GMIP*, 60(2), 1998.

[AHMP07] Umut A. Acar, Benoît Hudson, Gary L. Miller, and Todd Phillips. SVR: Practical engineering of a fast 3D meshing algorithm. In *Proc. 16th International Meshing Roundtable*, pages 45–62, 2007.

[BA76] Ivo Babuška and A. K. Aziz. On the Angle Condition in the Finite Element Method. *SIAM Journal on Numerical Analysis*, 13(2):214–226, April 1976.

[BEG94] Marshall Bern, David Eppstein, and John R. Gilbert. Provably Good Mesh Generation. *Journal of Computer and System Sciences*, 48(3):384–409, June 1994.

[BHV04] E. Boman, B. Hendrickson, and S. Vavasis. Solving elliptic finite element systems in near-linear time with support preconditioners. Archived by arxiv.org, cs.NA/0407022; in revision, SIAM J. Numer. Anal., 2004.

[BOG01] Charles Boivin and Carl F. Ollivier-Gooch. Guaranteed-quality simplicial mesh generation with cell size and grading control. *Engineering with Computers*, 17(3):269–286, 2001.

[BOG02] Charles Boivin and Carl Ollivier-Gooch. Guaranteed-Quality Triangular Mesh Generation for Domains with Curved Boundaries. *International Journal for Numerical Methods in Engineering*, 55(10):1185–1213, 20 August 2002.

[Bow81] Adrian Bowyer. Computing Dirichlet Tessellations. *Computer Journal*, 24(2):162–166, 1981.

[CDE+00] Siu-Wing Cheng, Tamal Krishna Dey, Herbert Edelsbrunner, Michael A. Facello, and Shang-Hua Teng. Sliver Exudation. *Journal of the ACM*, 47(5):883–904, September 2000.

[CDR07] Siu-Wing Cheng, Tamal K. Dey, and Edgar A. Ramos. Delaunay refinement for piecewise smooth complexes. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1096–1105, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[CDRR04] Siu-Wing Cheng, Tamal K. Dey, Edgar A. Ramos, and Tathagata Ray. Quality Meshing for Polyhedra with Small Angles. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, pages 290–299, Brooklyn, New York, June 2004. Association for Computing Machinery.

[Che89a] L. Paul Chew. Constrained Delaunay Triangulations. *Algorithmica*, 4(1):97–108, 1989.

[Che89b] L. Paul Chew. Guaranteed-quality triangular meshes. Technical Report TR-89-983, Department of Computer Science, Cornell University, 1989.

[Che97] L. Paul Chew. Guaranteed-Quality Delaunay Meshing in 3D. In *Proceedings of the Thirteenth Annual Symposium on Computational Geometry*, pages 391–393, Nice, France, June 1997. Association for Computing Machinery.

[Cia78] P. G. Ciarlet. *The finite element method for elliptic problems*. North-Holland, 1978.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT-press and McGraw-Hill, 2 edition, 2001.

[CP06] Siu-Wing Cheng and Sheung-Hung Poon. Three-dimensional delaunay mesh generation. *Discrete Comput. Geom*, 36:419–456, 2006.

[Del34] Boris Nikolaevich Delaunay. Sur la Sphère Vide. *Izvestia Akademia Nauk SSSR, VII Seria, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7:793–800, 1934.

[Dey06] Tamal K. Dey. *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*. Cambridge University Press, October 2006.

[Ede01] H. Edelsbrunner. *Geometry and Topology of Mesh Generation*. Cambridge Univ. Press, England, 2001.

[ELM+00a] Herbert Edelsbrunner, Xiang-Yang Li, Gary L. Miller, Andreas Stathopoulos, Dafna Talmor, Shang-Hua Teng, Alper Üngör, and Noel Walkington. Smoothing and cleaning up slivers. In *STOC*, pages 273–277, 2000.

[ELM+00b] Herbert Edelsbrunner, Xiang-Yang Li, Gary L. Miller, Andreas Stathopoulos, Dafna Talmor, Shang-Hua Teng, Alper Üngör, and Noel Walkington. Smoothing and cleaning up slivers. In *Proceedings of the 32th Annual ACM Symposium on Theory of Computing*, pages 273–277, Portland, Oregon, 2000.

[ES97] Herbert Edelsbrunner and Nimish R. Shah. Triangulating topological spaces. *IJCGA*, 7:365–378, 1997.

[HMP06] Benoît Hudson, Gary Miller, and Todd Phillips. Sparse Voronoi Refinement. In *Proceedings of the 15th International Meshing Roundtable*, pages 339–356, Birmingham, Alabama, 2006. Long version available as Carnegie Mellon University Technical Report CMU-CS-06-132.

[HMP07] Benoît Hudson, Gary L. Miller, and Todd Phillips. Sparse Parallel Delaunay Refinement. In *19th Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 339–347, San Diego, June 2007.

[HMPS09] Benoît Hudson, Gary L. Miller, Todd Phillips, and Donald Sheehy. Size complexity of volume meshes vs. surface meshes. In *19th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1041–1047, San Diego, January 2009.

[HPÜ05] Sariel Har-Peled and Alper Üngör. A Time-Optimal Delaunay Refinement Algorithm in Two Dimensions. In *Symposium on Computational Geometry*, 2005.

[Li03] Xiang-Yang Li. Generating well-shaped d-dimensional Delaunay meshes. *Theor. Comput. Sci.*, 296(1):145–165, 2003.

[LT01] Xiang-Yang Li and Shang-Hua Teng. Generating well-shaped Delaunay meshed in 3D. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 28–37. ACM Press, 2001.

[Mil04] Gary L. Miller. A time-efficient Delaunay refinement algorithm. In *Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 400–409, New Orleans, 2004.

[Moo95] Doug Moore. The cost of balancing generalized quadtrees. In *SMA '95: Proceedings of the Third Symposium on Solid Modeling and Applications*, pages 305–312, 1995.

[MPS07] Gary L. Miller, Todd Phillips, and Donald Sheehy. Size competitive meshing without large angles. In *34th International Colloquium on Automata, Languages and Programming*, 2007.

[MPS08] Gary L. Miller, Todd Phillips, and Donald Sheehy. Linear-size meshes. In *Canadian Conference on Computational Geometry*, 2008. To appear.

[MPV05] G. Miller, T. Phillips, and S. Vavasis. Solving laplace's equation. Personal Communication, 2005.

[MPW02a] Gary L. Miller, Steven E. Pav, and Noel J. Walkington. Fully incremental 3d Delaunay refinement mesh generation. In *Eleventh International Meshing Roundtable*, pages 75–86, 2002.

[MPW02b] Gary L. Miller, Steven E. Pav, and Noel J. Walkington. Fully Incremental 3D Delaunay Refinement Mesh Generation. In *Eleventh International Meshing Roundtable*, pages 75–86, Ithaca, New York, September 2002. Sandia National Laboratories.

[MTT+96] Gary L. Miller, Dafna Talmor, Shang-Hua Teng, Noel Walkington, and Han Wang. Control Volume Meshes Using Sphere Packing: Generation, Refinement and Coarsening. In *Fifth International Meshing Roundtable*, pages 47–61, Pittsburgh, Pennsylvania, October 1996.

[MTT99] Gary L. Miller, Dafna Talmor, and Shang-Hua Teng. Optimal coarsening of unstructured meshes. *Journal of Algorithms*, 31(1):29–65, Apr 1999.

[MTTW99] Gary L. Miller, Dafna Talmor, Shang-Hua Teng, and Noel Walkington. On the radius–edge condition in the control volume method. *SIAM J. Numer. Anal.*, 36(6):1690–1708, 1999.

[MV00] Scott A. Mitchell and Stephen A. Vavasis. Quality mesh generation in higher dimensions. *SIAM J. Comput.*, 29(4):1334–1370 (electronic), 2000.

[Pav03] Steven Elliot Pav. *Delaunay Refinement Algorithms*. PhD thesis, Department of Mathematics, Carnegie Mellon University, Pittsburgh, Pennsylvania, May 2003.

[PS85] Franco P. Preparata and Michael Ian Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, 1985.

[PW04] Steven E. Pav and Noel J. Walkington. Robust Three Dimensional Delaunay Refinement. In *Thirteenth International Meshing Roundtable*, pages 145–156, Williamsburg, Virginia, September 2004. Sandia National Laboratories.

[PW05] Steven E. Pav and Noel J. Walkington. Delaunay refinement by corner lopping. In *Fourteenth International Meshing Roundtable*, pages 145–156. Sandia National Laboratories, September 2005.

[Rup95] Jim Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms*, 18(3):548–585, 1995. Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) (Austin, TX, 1993).

[RW08] Alexander Rand and Noel Walkington. 3d delaunay refinement of sharp domains without a local feature size oracle. In *17th Internation Meshing Roundtable.*, 2008.

[SG05] Hang Si and Klaus Gärtner. Meshing Piecewise Linear Complexes by Constrained Delaunay Tetrahedralizations. In Byron W. Hanks, editor, *Proceedings of the Fourteenth International Meshing Roundtable*, pages 147–163, San Diego, California, September 2005.

[She97a] Jonathan Richard Shewchuk. *Delaunay Refinement Mesh Generation*. PhD thesis, Carnegie Mellon University, Pittsburgh, May 1997. CMU CS Tech Report CMU-CS-97-137.

[She97b] Jonathan Richard Shewchuk. *Delaunay Refinement Mesh Generation*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, May 1997. Available as Technical Report CMU-CS-97-137.

[She98] Jonathan Richard Shewchuk. Tetrahedral Mesh Generation by Delaunay Refinement. In *Proceedings of the Fourteenth Annual Symposium on Computational Geometry*, pages 86–95, Minneapolis, Minnesota, June 1998. Association for Computing Machinery.

[STÜ07] Daniel Spielman, Shang-Hua Teng, and Alper Üngör. Parallel Delaunay refinement: Algorithms and analyses. *IJCGA*, 17:1–30, 2007.

[Tal97] Dafna Talmor. *Well-Spaced Points for Numerical Methods*. PhD thesis, Carnegie Mellon University, Pittsburgh, August 1997. CMU CS Tech Report CMU-CS-97-164.

[Tan96] Tan. An optimal bound for high-quality conforming triangulations. *GEOMETRY: Discrete and Computational Geometry*, 15, 1996.

[Üng04] Alper Üngör. Off-centers: A new type of steiner points for computing size-optimal guaranteed-quality delaunay triangulations. In *Proceedings of LATIN*, 2004.