Robust, Guaranteed-Quality Anisotropic Mesh Generation

by Jessica Schoen

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Jonathan Shewchuk
Research Advisor

(Date)

* * * * * * *

Professor Satish Rao
Second Reader

(Date)

Contents

Chapter 1

Introduction

Despite years of study, mesh generation remains a bottleneck in many finite element simulations involving complex shapes. The difficultly arises in part because generating a high quality mesh depends not only on the geometry but also on the equations governing the simulation themselves. Under the right circumstances, *anisotropic* meshes can greatly reduce the number of elements, and therefore the computation time, required for simulations. In computer graphics, anisotropic meshes can accomplish similar goals for rendering. This work presents an algorithm for generating two-dimensional anisotropic meshes with a guarantee on the quality of their elements.

Anisotropic meshes contain long, thin elements that have been stretched according to a preferred direction inherent to a specific problem. Although most mesh generation algorithms endeavor to avoid creating such triangles, "long and thin triangles can be good for linear interpolation," as Rippa observes in his aptly titled paper [22]. He finds that, for the linear interpolation of a function F over a triangulation, "triangles should be long in directions where the magnitude of the second directional derivative of F is small and thin in directions where the magnitude of the second directional derivative of F is large."

Rippa studied the error in the value of the interpolated function, but that may not be one's only concern. For some applications, including rendering, the error in the *gradient* of the interpolated solution can be even more important. With both of these considerations in mind, Apel [2] and Shewchuk [23] provide in-depth looks at the benefits of anisotropic elements. Simulation problems for which anisotropic meshes are appropriate are those whose solutions have different behaviors in different directions. Types of simulations with this property include "boundary layers in viscous flow problems and in various plate and shell models, shock phenomena in flow problems, and singularities near edges in Poisson type problems like diffusion and linear elasticity" [2]. In computer graphics, anisotropic meshes can be beneficial for rendering surfaces with differing curvatures in different directions.

1.1 Anisotropy

For two-dimensional problems, anisotropy is often represented by a 2-by-2 symmetric positive definite matrix called a *metric tensor* at each point of the domain. In practice the metric tensor might only be known at discrete points, such as at the vertices of a background mesh or a mesh from a previous timestep. In these cases, a metric tensor can be estimated at a point where it is not explicitly given through interpolation.

A geometric interpretation of the metric tensor M_p at a point p is that it describes how distances and angles are measured from p's point of view. As in the work of Labelle and Shewchuk [17], for a 2-by-2 metric tensor M_p, define a *deformation tensor* F_p to be any 2-by-2 matrix such that

$$M_p = F_p^T F_p \qquad \text{and} \qquad \det\, F_p > 0.$$

Then F_p maps the physical domain to a warped space where distances and angles can be measured as they are seen by p simply by measuring them in the usual Euclidean way. See Fig. 1.1, which is from the paper by Labelle and Shewchuk. In the figure, thin arcs around p represent *isocontours*. The isocontours of p are sets of points that are equidistant from p when measured from p's point of view. Likewise, the thin arcs around q are q's isocontours. The heavy arc is a *Voronoi arc* — the set of points at the same distance from p and q.

1.2 Metric Tensors

For the purposes of the anisotropic mesh generation algorithm given here, assume that the metric tensor field is provided as user input. Let us consider how these metrics may be generated in practice.

Meshes used in finite element simulations commonly use solution-driven adaptivity; the metric tensor field is generated from the Hessian of a numerically computed solution. With this approach, how to generate the initial mesh when no numerical solution has been computed yet is not obvious. In some cases, the user may have *a priori* knowledge of the behavior of the Hessian. Otherwise, a coarse isotropic mesh may be used to get a rough estimate of the solution. Using the Hessian of this approximate solution as a metric, a finer anisotropic mesh can then be generated. The simulation then begins in earnest with the fine mesh.

The idea of using the Hessian as a metric in was popularized by results of D'Azevedo [11] and D'Azevedo and Simpson [12] on the linear interpolation error in a function and its gradient, respectively. In an unusual application, Courchesne *et al.* use the Hessian to generate meshes of

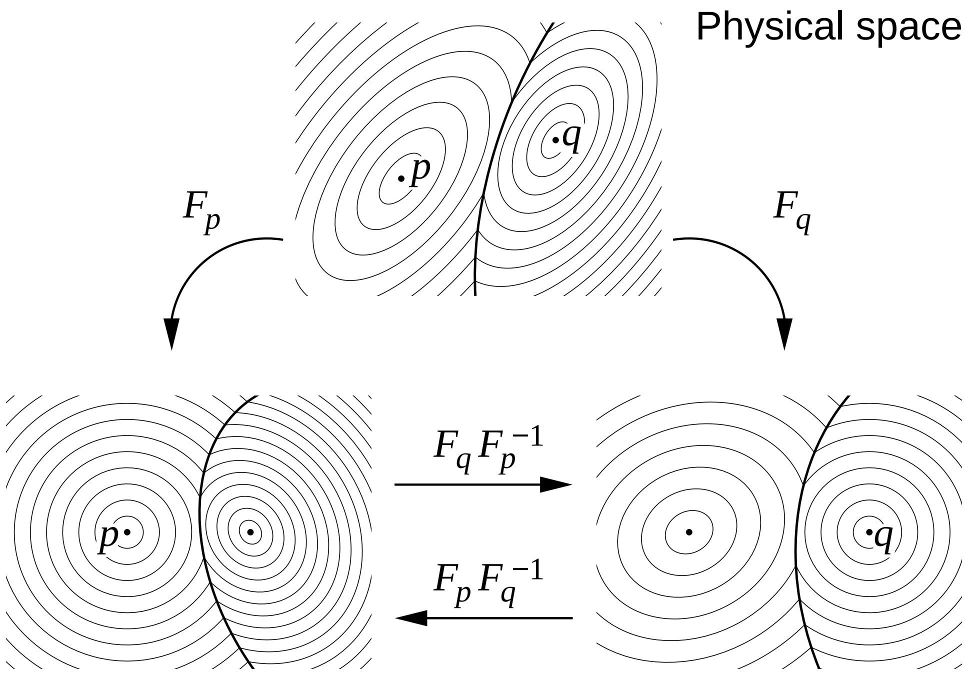

Figure 1.1: The deformation tensors F_p and F_q transform the domain according to the metric tensors at p and q, respectively. The elliptical isocontours around p in the top row become circular when space is warped according to F_p (lower left), and likewise for q (lower right).

MRI images [10]. They define a function over the pixels of an MRI scan whose value at each pixel is equal to the intensity of the image at that pixel. Then they use the Hessian of this function as a metric for anisotropic mesh generation.

In general, a pitfall of using the Hessian matrix to define a metric tensor field is that the Hessian of the true solution to the problem is not known. An approximate Hessian must be "recovered" from a numerical solution that has been computed. A variety of techniques for Hessian recovery have been studied; see the work of Buscaglia and Dari [8], for example.

Another downside of generating a metric from the Hessian is that, depending on the mesh generation algorithm, it may result in a mesh with neighboring elements of widely varying sizes. Such poor grading is unacceptable in some engineering applications. To combat this difficulty, Li *et al.* [18] give an anisotropy-preserving procedure to run before the mesh is generated, which smooths out the metric tensor field. Their goal is to improve the grading of the mesh and avoid poor quality elements that result from rapid changes in the metric tensors of adjacent elements.

Constructing a metric tensor field from the Hessian of a function is not the only option. Gruau and Coupez [15] define two different metrics that they find especially useful for material processing. The first allows the user to specify the number of layers of elements throughout the thickness of the domain. Their second metric is concerned with interfaces between different materials. A common method for handling such interfaces is to use a constrained Delaunay triangulation, which constrains the mesh to include the edges of the interface. Gruau and Coupez instead define a metric which causes the mesh to be very refined along the interface; see Fig. 1.2. They prefer to use this metric with an anisotropic mesh generator rather than a constrained Delaunay triangulation because they find that it simplifies the remeshing steps of their finite element simulations.

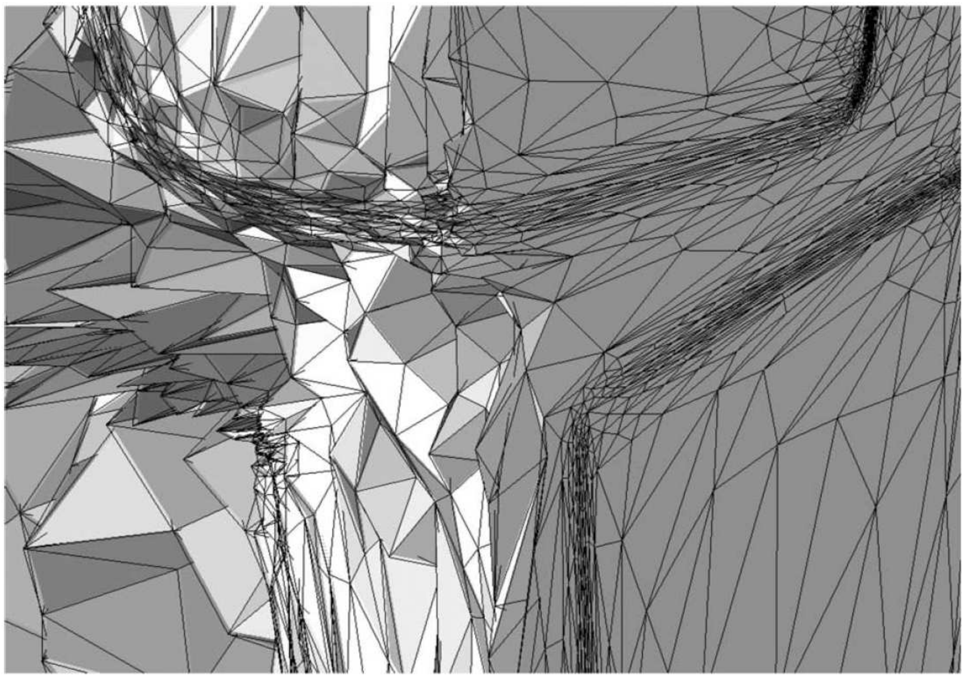

Figure 1.2: Gruau and Coupez use a metric tensor field that causes the mesh to be very refined along interfaces between materials.

For surface meshing, Alliez *et al.* [1] and George and Borouchaki [14] among others, choose to define metrics based on the curvature of a surface. Alliez *et al.* find that this choice of metric produces visually pleasing results. George and Borouchaki show that it allows them to generate a mesh that approximates the surface well.

Given the variety of options for defining a metric tensor field for mesh generation, researchers have also devised ways of studying metrics themselves. Tchon *et al.* [25] develop a method for visualizing metrics which is independent from any mesh generation algorithm; see Fig. 1.3. Lipnikov and Vassilevski [20] derive error estimates which allow them to study the effects of different metric modifications (such as the smoothing of Li *et al.* [18]) on the interpolation error.

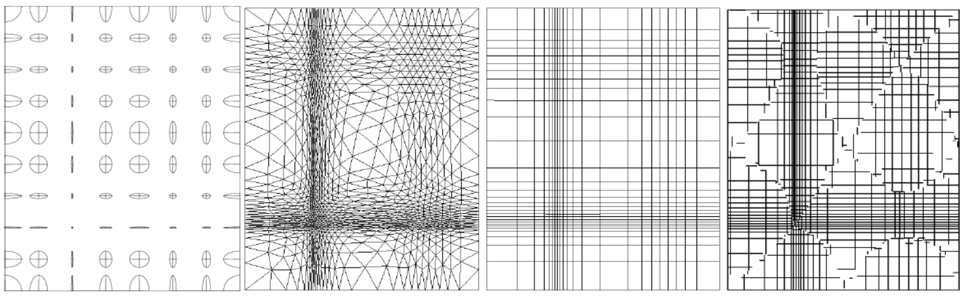

Figure 1.3: Tchon *et al.* compare their metric visualization technique (far right) with a more typical metric visualization (far left) and triangular (second from left) and quadrilateral (second from right) meshes based on the metric.

Chapter 2

Previous Work

2.1 Heuristic Approaches

The anisotropic mesh generation problem has been tackled with a variety of heuristic approaches. In chronological order, summaries of these techniques are given here.

The anisotropic mesh generation algorithm described by Bossen and Heckbert [6] is a generalization of an algorithm for isotropic mesh generation given in the same paper. Their isotropic algorithm is what they refer to as a "pliant" mesh generation algorithm, one "in which smoothing, insertion, and deletion take place in a loop."

The isotropic algorithm first creates the constrained Delaunay triangulation of the input and marks all of the vertices as active. It then proceeds iteratively by randomly choosing an active vertex to reposition based on the locations of its neighbors. Neighbors that are too close repel the vertex, and neighbors that are too far away attract it. If the vertex is not moved too far, it is marked inactive. The vertices are then retriangulated so that the mesh is still constrained Delaunay. If the density of the mesh around the repositioned vertex is too high, the vertex is deleted. Otherwise, a new vertex is inserted. The algorithm continues until all of the vertices are inactive. However, an inserted or deleted vertex causes all of its neighbors to become active, so the status of a vertex may flip-flop several times during the course of the algorithm.

Bossen and Heckbert transform this algorithm into an anisotropic mesh generation algorithm by replacing the Delaunay circumcircle test with an anisotropic version. To determine whether to flip the edge between two triangles, the metric tensors at the four vertices of the triangles are averaged, with an equal weight given to each vertex. The circumcircle test is performed with distances and angles measured according to this average metric tensor. This algorithm is not guaranteed to converge, although the authors only had difficulties on inputs with abrupt changes in the metric tensor or when the user requested large edge lengths near small input features. They subjectively

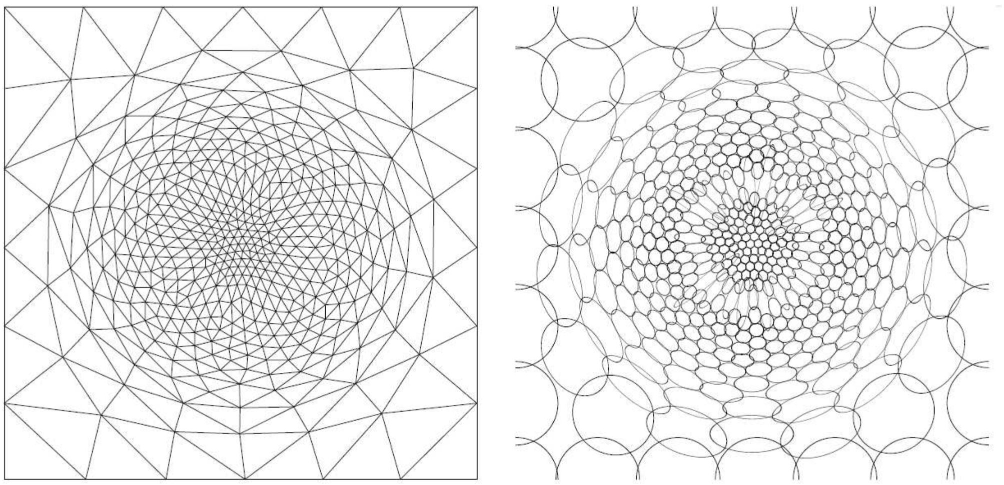
Figure 2.1: Left: An anisotropic mesh generated by Bossen and Heckbert's algorithm. Right: Ellipses whose centers are the vertices of the mesh. Each ellipse represents the set of points at a distance of 1 from its center as measured by the metric tensor at the center.

found their output meshes "attractive," but could make no guarantee on their quality. See Fig. 2.1 for an example of a mesh generated by this algorithm.

Borouchaki, Frey, and George [4], in the same conference as Bossen and Heckbert, introduced a different approach to the anisotropic mesh generation problem. The details of their method can be found in later works by Borouchaki, George, and other authors [5, 14]. The method is an extention of the Bowyer–Watson Delaunay triangulation algorithm [7, 26] to the anisotropic case. The original Bowyer–Watson algorithm creates a sequence of Delaunay triangulations by adding vertices one at a time. To add a vertex to the triangulation, all of the triangles whose circumcircles contain the vertex are removed, leaving a star-shaped cavity that contains the new vertex. The cavity is then triangulated by adding an edge between each of its vertices and the new vertex.

Borouchaki *et al.* call this process of advancing from one triangulation to the next the "Delaunay kernel," and they propose three anisotropic Delaunay kernels. The first computes the cavity based only on the metric tensor at the vertex that is currently being added to the triangulation. The second determines if a given triangle belongs in the cavity using the metric tensors at both the new vertex and at one of the triangle's vertices. The third uses the tensors at the new vertex and at all three of the vertices of the triangle. The authors prefer their second option because they did not notice a "significant difference" between the second and third kernels, and the second requires about half of the computation needed for the third. Provided that one uses a depth first search to find the triangles that form the cavity, they prove that all three kernels result in valid triangulations in two dimensions. This result does not hold in higher dimensions, and there is no guarantee on the quality of the mesh in any dimension.

The ellipsoidal bubble packing method of Shimada, Yamada, and Itoh [24] is similar to Bossen and Heckbert's method in that both are iterative schemes that reposition elements based on the locations of their neighbors. In the bubble packing method, the authors define a "proximity-based interbubble force" based on the van der Waals force. Although it is based on this physical force, which implies that it can both attract and repel bubbles, the interbubble force is simplified and quicker to compute than the true van der Waals force. One of the main differences between this algorithm and that of Bossen and Heckbert is that here bubbles are packed in order of dimension: first on vertices, then on edges, and finally on faces. Also, Shimada *et al.* work on parametric surfaces in three dimensions, while Bossen and Heckbert only implemented their technique for two-dimensional domains.

The ellipsoidal bubbles are packed by finding the forces acting on each bubble and then solving the resulting system of equations of motion for a force-balancing configuration. The positions of the bubbles are updated and bubbles are added or removed if an area is too sparsely or densely packed. This process is repeated until the bubbles no longer move very far when their position is updated.

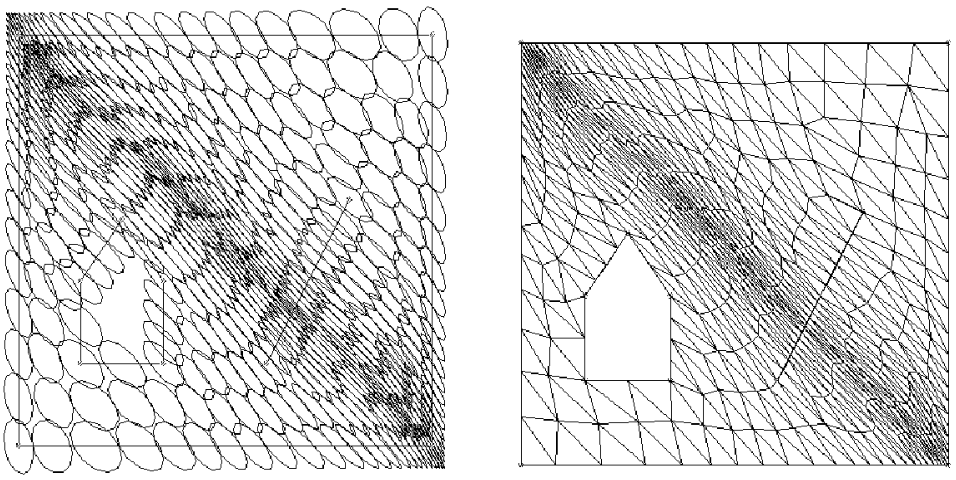

Figure 2.2: Left: A bubble packing generated by the algorithm of Shimada *et al.* The bubbles are allowed to overlap significantly. Right: The corresponding mesh.

Once the number of bubbles and their positions are finalized, their centers are triangulated. Shimada *et al.* use an anisotropic version of the Delaunay circumcircle test, though not the same version as that used by Bossen and Heckbert. Given four vertices that form a convex quadrilateral, the authors perform the standard circumcircle test in the space given by the metric tensor at the barycenter of the quadrilateral. Fig. 2.2 gives an example of a mesh generated with this algorithm.

The authors found that this method produced "high quality" meshes in practice, but do not provide guarantees on mesh quality. They also determined that the initial distribution of bubbles played a large role in the number of iterations required for the method to converge.

Li, Teng, and Üngör combine advancing front and ellipse packing schemes in their anisotropic mesh generation algorithm [19]. Like the previous algorithms, this one consists of a vertex generation phase followed by a triangulation phase.

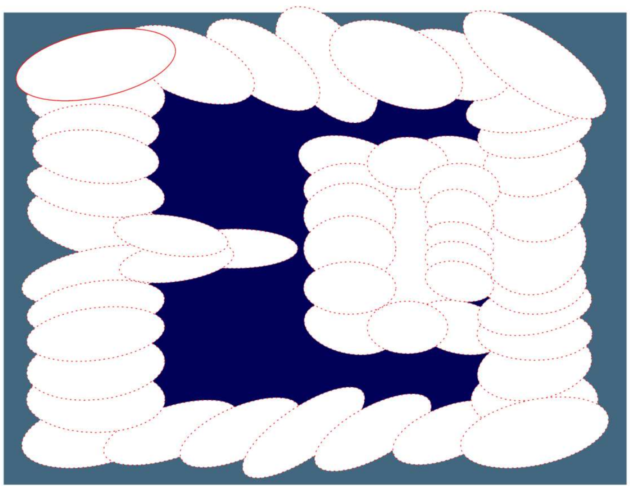

Figure 2.3: Li *et al.*'s advancing front after biting one "layer" of ellipses.

First, all of the vertices of the mesh are generated by packing ellipses along an advancing front. A key element of the algorithm is the "biting ellipse" of a vertex, which is the set of all points that are within a predetermined constant distance from the vertex, as measured by the metric tensor at the vertex. When a vertex is inserted, the boundary arc of its biting ellipse is added to the advancing front, see Fig. 2.3. The vertices are then triangulated using a modified Delaunay edge-flip test which takes anisotropy into account. The test used here is different from than that used by Bossen and Heckbert, as well as that of Shimada *et al.*

Li *et al.* prove that their algorithm generates a "weak-β-ellipse-packing," which implies that the ellipses are neither too densely nor too sparsely packed. The algorithm also always terminates and produces a valid mesh. Although the authors suggest measuring the quality of each triangle in the mesh based on the metric tensor at the triangle's barycenter, there is no guarantee on the quality of the output mesh. Furthermore, the algorithm given in the paper is not complete — the authors avoid the difficult question of how to maintain the advancing front. The algorithm also has never been implemented.

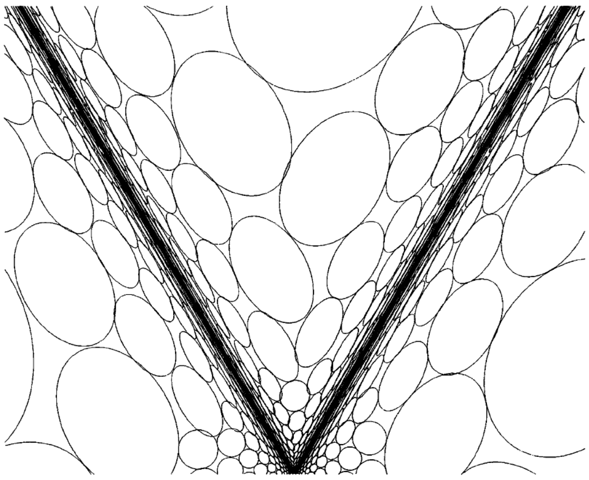

Figure 2.4: An ellipse packing generated by Lo and Wang. The ellipses overlap only slightly.

The method of Lo and Wang [21] is similar to the other ellipse packing methods. Their contribution is an algorithm for unbounded domains. Rather than initially packing ellipses on the domain's boundary, they begin at an arbitrarily chosen point in the interior of the domain and grow their ellipse packing radially outward. Once the front is far enough away from the interesting features of the domain or once a user-defined number of ellipses have been created, the algorithm terminates. See Fig. 2.4 for an ellipse packing created by this algorithm. As with the other ellipse-based schemes, they generate satisfactory meshes in practice, but provide no guarantees.

Alliez *et al.* [1] focus on anisotropically remeshing scanned surfaces, with the anisotropy dictated by the surface's two principal directions of curvature. Their technique attempts to mimic the curvature lines that an artist would naturally draw. The resulting mesh contains stretched quadrilaterals in anisotropic regions and triangles in isotropic regions.

The remeshing algorithm first approximates the principal curvatures of the surface. Then a discrete conformal parameterization is used to flatten the curvature tensor field from the three-dimensional surface to a two-dimensional domain. Next, a Gaussian filter smoothes the two-dimensional tensor field, and vertices where the tensor is roughly isotropic are flagged for special treatment later. The algorithm then uses the principal curvatures to create a network of lines on the surface. The interchapters of these lines, along with additional vertices in the isotropic regions, become the vertices in the final quad-dominant mesh.

The algorithm is robust, but guarantees on the quality of the final meshes are not available. The authors did successfully generate a variety of visually appealing meshes of complex surfaces, including the heads of the Stanford bunny, see Fig. 2.5, and Michelangelo's David. Several global

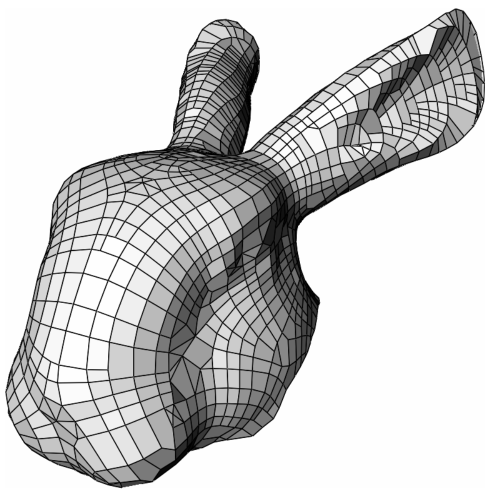

Figure 2.5: The head of the Stanford bunny, meshed with the algorithm of Alliez *et al.*

properties of the output mesh can be dictated by the user, including the mesh density, the amount of anisotropy, and the degree to which curvature of the surface is reflected in the mesh.

There are a number of other anisotropic meshing techniques that will not be discussed in detail. From before Bossen and Heckbert's paper through the present, anisotropic mesh modification algorithms have appeared in the engineering literature [8, 13, 15]. These algorithms differ from the anisotropic mesh generation algorithms in this chapter because their input is a preexisting mesh which they modify to make anisotropic. The input mesh is often an isotropic mesh or a mesh from the previous timestep of a simulation. The basic strategy of these algorithms is to make local changes to the input mesh, such as edge flipping, edge collapsing, and vertex relocation, until the mesh conforms to the metric tensor field "well enough." Haimes and Aftosmis [16] take a different approach in their algorithm, which generates structured anisotropic meshes for a restricted class of CAD objects.

2.2 Guaranteed-Quality Anisotropic Mesh Generation

The first guaranteed-quality anisotropic mesh generation algorithm was given by Labelle and Shewchuk [17]. This algorithm is based on the anisotropic Voronoi diagrams defined in the same paper; see Fig. 2.6. The authors give conditions under which the anisotropic Voronoi diagram is dual to a proper triangulation. They also prove that their algorithm generates a mesh in which no triangle contains an angle smaller than 20° when measured with the metric tensor of *any* point in the triangle. This proof is especially significant in light of the fact that no earlier anisotropic mesh generation algorithm came with any guarantee on the quality of the mesh it produced, and termination was

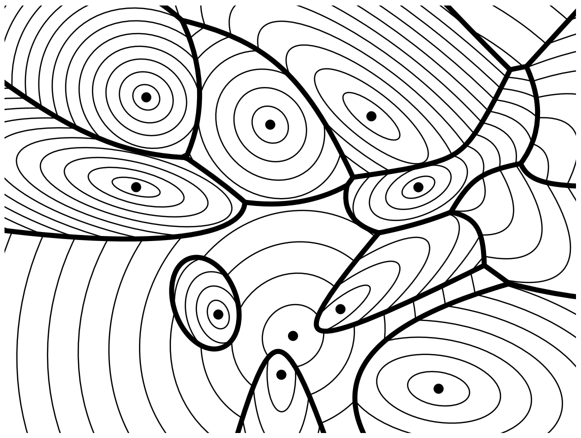

Figure 2.6: An anisotropic Voronoi diagram. The bold curves are boundaries of the Voronoi cells, and the light curves are isocontours.

not guaranteed for many of the iterative schemes. The vertices of an anisotropic Voronoi diagram are interchapter points of conic chapters, and Labelle and Shewchuk's algorithm is vulnerable to the robustness of the computation of these interchapters. This vulnerability prevents an implementation of this algorithm from being both fast and robust.

In addition to the present thesis, a number of others which are based on the work of Labelle and Shewchuk have appeared. For instance, Boissonnat *et al.* [3] construct the d-dimensional anisotropic Voronoi diagram by computing a power diagram in $(d+1)$-dimensions. Although their approach works in any dimension, they have proven only that their corresponding mesh generation algorithm terminates in two dimensions. Cheng and co-authors [9] apply the work of Labelle and Shewchuk in their three-dimensional anisotropic surface meshing algorithm.

Chapter 3

Star-Based Anisotropic Mesh Generation Algorithm

As the name implies, the basic building block of this algorithm is a star.

Definition 1 *Given a triangulation T, the* star *of a vertex v is the set of all simplices (vertices, edges, faces, and triangles) in T having v for a face. See Fig. 3.1.*

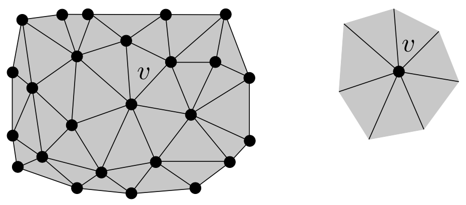

Figure 3.1: On the left, a triangulation containing vertex v. On the right, star(v) in that triangulation.

One of the main ideas in this algorithm is that each site computes its own star independently of all the other sites. Performing local computations rather than constructing the global anisotropic Voronoi diagram allows us to avoid some of the numerical difficulties that Labelle and Shewchuk [17] encountered. However, it also introduces a challenge: refinement is required to ensure that the stars are consistent with each other.

3.1 Inconsistent Stars

A naïve approach to anisotropic star construction demonstrates how two stars may be inconsistent. Suppose the set of sites to be triangulated consists of four sites, a, b, c, and d, with isocontours at c and d as shown in Fig. 3.2. To compute the star of c, one could imagine doing the following:

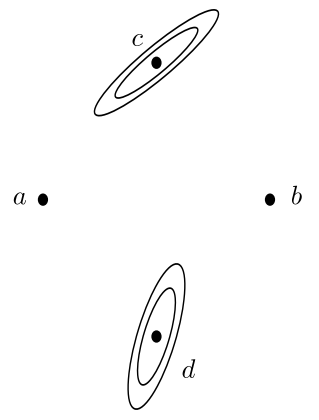

Figure 3.2: Sites a, b, c, and d, with isocontours shown at c and d.

1. Apply F_c to warp the positions of all the sites according to c's perspective
2. Compute the Delaunay triangulation of the warped sites
3. Apply F_c^{-1} to the triangulation in order to undo the warping
4. Find star(c) in this triangulation

Likewise for d's star, the process could be repeated, replacing F_c with F_d and c with d. This procedure results in the warped Delaunay triangulations shown on the left side of Fig. 3.3 and the corresponding stars shown on the right side of the same figure. These stars are *inconsistent* because edge $\overline{cd}$ appears in d's star but not in c's star.

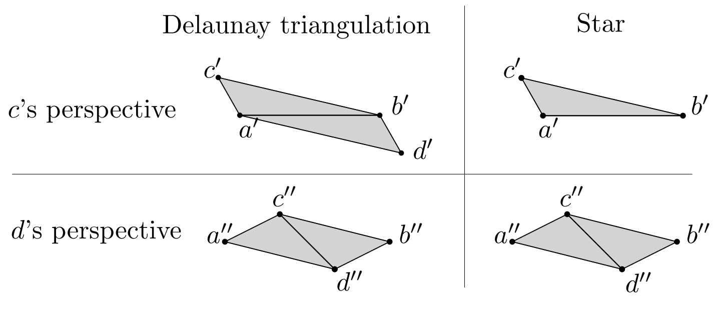

Figure 3.3: Points a', b', c', d' represent the location of sites a, b, c, d as viewed by c. Similarly, a'', b'', c'', d'' represent the original four sites as seen from d's perspective.

The star construction algorithm in section 3.3 guarantees that all four sites will always make the same decision about which edge to include in their stars if they are faced with the same choice.

3.2 Key Facts From Prior Work

The following definitions, lemmas, and theorems from the work of Labelle and Shewchuk [17] are also used here and are stated without proof.

The distance between two points q_1 and q_2 as measured by p is

$$d_p(q_1, q_2) = ||F_p q_1 - F_p q_2||_2 = \sqrt{(q_1 - q_2)^T M_p (q_1 - q_2)}.$$

The shorthand notations $d_p(q) = d_p(p, q)$ and $d(p, q) = \min\{d_p(q), d_q(p)\}$ are also used.

Definition 2 ***(Anisotropic Voronoi diagram)*** *Let V be a set of sites. The* Voronoi cell *of a site v in V is*

$$\mathrm{Vor}(v) = \{p \in E^d : d_v(p) \le d_w(p) \,\text{for all}\, w \in V\}.$$

Any subset of sites $W \subseteq V$ induces a Voronoi cell $\mathrm{Vor}(W) = \cap_{w \in W} \mathrm{Vor}(w)$ of points equally close to the sites in W and no closer to any others. If it is not empty, such a cell has dimensionality of $\dim(\mathrm{Vor}(W)) \ge d + 1 - |W|$, achieving equality if the sites are in general position. Every site in W is said to own *$\mathrm{Vor}(W)$. The* anisotropic Voronoi diagram *of V is the arrangement of the Voronoi cells $\{\mathrm{Vor}(W) : W \subseteq V, W \ne \emptyset, \mathrm{Vor}(W) \ne \emptyset\}$.*

While the dual of the standard Voronoi diagram is the Delaunay triangulation of the sites, the dual of an anisotropic Voronoi diagram may not be a triangulation at all.

Definition 3 *Let v and w be two sites. Define the* wedge *between these two sites as the locus of points q for which the angle $\angle qvw$ as viewed from v is less than $90°$, and the angle $\angle qwv$ as viewed from w is less than $90°$. (See Fig. 3.4.) Mathematically,*

$$\mathrm{wedge}(v, w) = \{q \in E^d : (q - v)^T M_v (w - v) > 0 \quad \text{and} \quad (q - w)^T M_w (v - w) > 0\}.$$

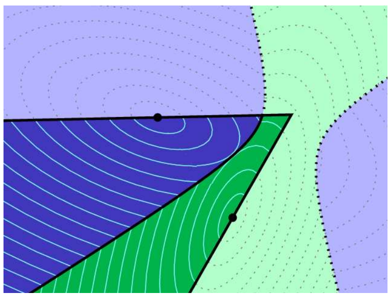

Figure 3.4: The wedge of two sites.

Using wedges, Labelle and Shewchuk proved many properties of anisotropic Voronoi diagrams.

Lemma 1 ***(Visibility Lemma)*** *Let v and w be two sites in E^d. If we restrict the two-site Voronoi diagram of $\{v, w\}$ to* wedge(v, w)*, then v can see its entire cell, and w can see its entire cell as well.*

Theorem 2 ***(Visibility Theorem)*** *If every lower-dimensional face of a d-face of* Vor(v) *is wedged, then the d-face is star-shaped and every point in the d-face is visible from v.*

Lemma 3 ***(Triangle Orientation Lemma)*** *Let q be a Voronoi vertex owned by the sites v_1, v_2, v_3. If q is wedged, then the orientation of the triangle $v_1v_2v_3$ matches the ordering of the cells* $\mathrm{Vor}(v_1)$, $\mathrm{Vor}(v_2)$, $\mathrm{Vor}(v_3)$ *locally around q. In other words, if at q the cells* $\mathrm{Vor}(v_1), \mathrm{Vor}(v_2), \mathrm{Vor}(v_3)$ *occur clockwise, then the sites v_1, v_2, v_3 occur clockwise in the plane, and vice versa.*

Wedges are also used in the Dual Triangulation Theorem, which characterizes anisotropic Voronoi diagrams that dualize to proper triangulations.

Theorem 4 ***(Dual Triangulation Theorem)*** *Let the domain Ω be a polygonal subset of the plane, let V be a set of sites in Ω that includes every vertex of Ω, and let D be the anisotropic Voronoi diagram of V. Let $D|_\Omega$ be the restriction of D to Ω. Suppose that each Voronoi arc cut by the restriction operation is owned by the endpoints of the edge of Ω that cuts it. If all the Voronoi arcs and vertices of $D|_\Omega$ are wedged, then the geometric dual of $D|_\Omega$ is a polygonalization of Ω (with strictly convex polygons), and is a triangulation of Ω if V is in general position. Arbitrarily triangulating each polygon yields what we call an* anisotropic Delaunay triangulation *of (V, Ω).*

3.3 Incremental Insertion Algorithm for Constructing a Single Star

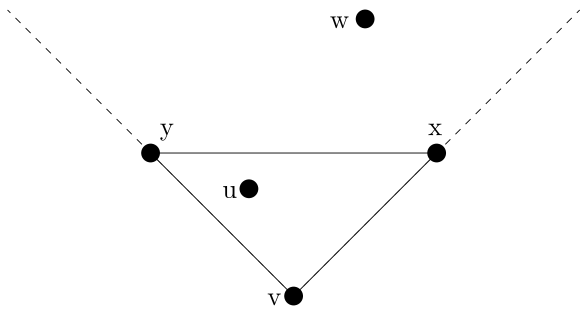

Figure 3.5: u and w subtend $\triangle vxy$

Given a set S of sites, the star of a site v is constructed by considering all of the other sites one at a time and inserting them into v's star with the InsertSite routine if they pass the InsertionTest. Sites that have been inserted into v's star may be removed at a later time, but each site will be inserted at most once.

Definition 4 *A site w* subtends *triangle $\triangle vxy \in$* star(v) *if w lies between the rays $\overrightarrow{vy}$ and $\overrightarrow{vx}$. See Fig. 3.5.*

Definition 5 *The* allowable region *for sites w and v when w subtends $\triangle vxy$ is the set of all points p in the plane such that $x \notin \triangle vwp$ and $y \notin \triangle vwp$. See Fig. 3.6.*

Definition 6 *Site v* dominates *point p over site w if $|\overline{vp} \cap \text{Vor}\{v, w\}| < |\overline{wp} \cap \text{Vor}\{v, w\}|$ in the two-site Voronoi diagram of $\{v, w\}$. See Fig. 3.7.*

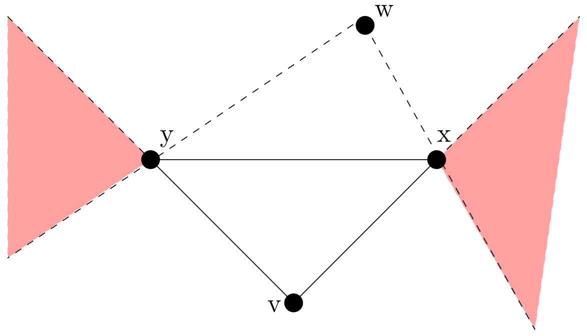

Figure 3.6: The allowable region for w and v consists of the entire plane except for the shaded regions.

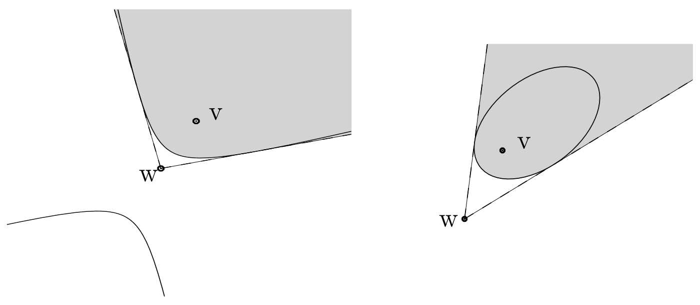

Figure 3.7: In both cases, v dominates the shaded regions over w

The incremental insertion algorithm is driven by two subroutines, InsertionTest and InsertSite.

```
ConstructStar($v, S$) :
    star($v$) ← empty star
    for each site $s \in S$
        if (InsertionTest($s$, star($v$)) returns Yes
            InsertSite($s$, star($v$))
```

```
INSERTIONTEST($w$, star($v$)) :
    if ($w$ is colinear with $v$ and a site $u \in$ star($v$) && $u$ lies between $w$ and $v$)
        return NO
    else if ($w$ is colinear with $v$ and a site $u \in$ star($v$) && $w$ lies between $u$ and $v$)
        remove $u$ from star($v$)
    else if ($w$ is strictly inside a triangle in star($v$))
        return YES

    if (there is a triangle $\triangle vxy \in$ star($v$) that $w$ subtends)
        $\mathcal{C} \leftarrow$ the portion of the arc Vor($\{w, v\}$) that is visible to both $w$ and $v$
            and that lies within the allowable region for $w$ and $v$
        if (there is a point on $\mathcal{C}$ that $w$ dominates over $x$ and $y$ and that $v$ dominates over $x$ and $y$)
            return YES
        else
            return NO
    else
        return YES
```

```
INSERTSITE($w$, star($v$)) :
    Add $w$ to star($v$)
    if (star($v$) contains fewer than 3 sites (other than $v$))
        DONE
    Let $x$ and $y$ be $w$'s neighbors in star($v$)
    while (INSERTIONTEST($x$, star($v$)) returns NO)
        $z \leftarrow x$'s neighbor in star($v$) other than $w$
        remove $x$ from star($v$)
        $x \leftarrow z$
    while (INSERTIONTEST($y$, star($v$)) returns NO)
        $z \leftarrow y$'s neighbor in star($v$) other than $w$
        remove $y$ from star($v$)
        $y \leftarrow z$
```

3.4 Equivalence Theorem

If the spacing of the sites is dense enough, then the independently constructed star of v assembled using the above algorithm contains the same sites as star(v) in the anisotropic Voronoi diagram of V. Before proving this equivalence, we need a few lemmas and facts.

Lemma 5 *Let w, v, and y be sites of an anisotropic Voronoi diagram for which $y \in \triangle wvp$ for some point p on* Vor($\{v, w\}$)*. If the ray $\overrightarrow{py}$ first passes through* Vor(v) *or* Vor(w)*, then the Voronoi arcs and vertices cannot all be wedged.*

Proof: Assume that the Voronoi arcs and vertices are all wedged and seek a contradiction. Also assume, without loss of generality, that w, v, p are in clockwise order.

There may be more than one site inside of $\triangle wvp$; choose which site to call y in the following way: If there is only 1 site in $\triangle wvp$, let y be that site. Otherwise, let y be the site such that sites w, y, u are in counterclockwise order for all sites $u \in \triangle wvp$. There may be multiple sites on a common line through w which satisfy this criterion. In this case, let y be the one farthest from w. See Fig. 3.8.

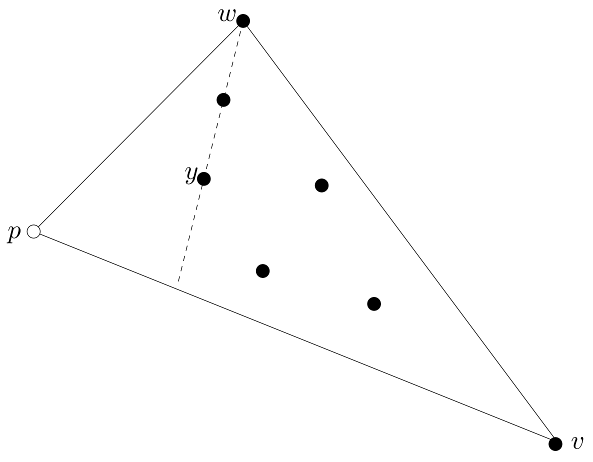

Figure 3.8: If there is more than one site in $\triangle wvp$, choose y to be the site for which w, y, u are in counterclockwise order for all $u \in \triangle wvp$ and that is farthest from w in the case of colinearities.

Let k be the number of Voronoi cells intersected by the line segment $\overline{py}$. Note that k must be at least 2 because $\overline{py}$ begins in $\mathrm{Vor}(w)$ or $\mathrm{Vor}(v)$ and ends in $\mathrm{Vor}(y)$. Assume that the sites have been perturbed so that $\overline{py}$ does not intersect any Voronoi vertices. Now proceed by induction.

Base case: $k = 2$. Two cases are considered, one in which $\overline{py}$ intersects $\mathrm{Vor}(w)$ and $\mathrm{Vor}(y)$, and the other in which $\overline{py}$ intersects $\mathrm{Vor}(v)$ and $\mathrm{Vor}(y)$.

Case I: First suppose that $\overline{py}$ intersects $\mathrm{Vor}(w)$ and $\mathrm{Vor}(y)$. Let s be the point where $\overrightarrow{wy}$ intersects $\overline{pv}$.

Note that $\overline{py}$ must intersect $\mathrm{Vor}(\{w, y\})$. Imagine following $\mathrm{Vor}(\{w, y\})$ from its point of intersection with $\overline{py}$ away from w. The Visibility Theorem guarantees that this arc

- will not intersect $\overline{py}$ again, because the intersection point closest to y would block y's view of any other intersection points on $\overline{py}$,
- will not intersect $\overline{ps} \subset \overline{pv}$ since all of $\overline{pv}$ is in $\mathrm{Vor}(v)$,
- will not intersect $\overline{ys}$, since y would block w's view of the intersection,
- is not an ellipse.

So, moving from the arc's intersection with $\overline{py}$ away from w, the arc does not leave $\triangle pys$. Since the arc is also not an ellipse, there is a Voronoi vertex z on $\mathrm{Vor}(\{w, y\})$ in $\triangle pys$.

Let u be the site other than w and y that owns z. All of $\mathrm{Vor}(\{w, y\})$ must be visible to both w and y, and $\overline{uz} \subseteq \mathrm{Vor}(u)$. These two observations imply that z sees the Voronoi *cells* of w, y, u in clockwise order.

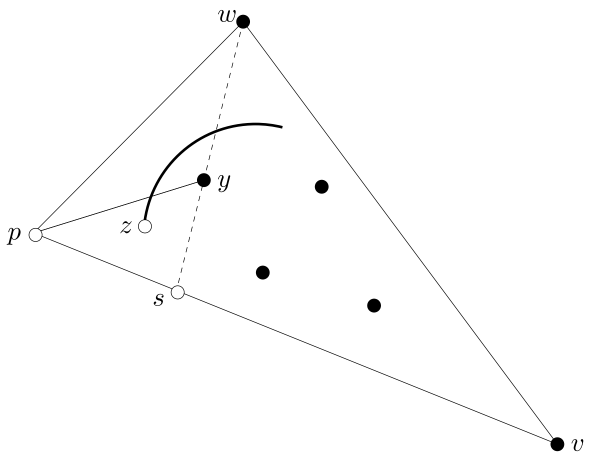

Figure 3.9: z is the Voronoi vertex of w, y, and another unknown site u. $s = \overrightarrow{wy} \cap \overline{pv}$.

By the Triangle Orientation Lemma, if the sites w, y, u are in counterclockwise order or are colinear, then z is not wedged and the lemma holds. So suppose that the sites w, y, u are in clockwise order. Then since u must be able to see z, u may not lie outside of $\triangle wvp$. This is because $\overline{uz}$ would have to intersect either $\overline{pw}$ or $\overline{pv}$, which are entirely contained in $\mathrm{Vor}(w)$ and $\mathrm{Vor}(v)$, respectively. The site u must therefore be in $\triangle wsp$. But y was chosen so that the sites w, y, u are in counterclockwise order or are colinear for all sites $u \in \triangle wvp$, and the desired contradiction has been found.

Case II: Now suppose that $\overline{py}$ intersects $\mathrm{Vor}(v)$ and $\mathrm{Vor}(y)$. Let r be the point where $\overrightarrow{vy}$ intersects $\overline{pw}$. Let z be the Voronoi vertex on $\mathrm{Vor}(\{v, y\})$ that lies on the opposite side of $\overline{py}$ from v. By the same reasoning as above, z exists and is in $\triangle pry$. Let u be the site other than v and y that owns z. The Voronoi cells of y, v, u are in clockwise order around z.

If the sites y, v, u are in counterclockwise order or are colinear, then z is not wedged and the lemma holds. So suppose that the sites y, v, u are in clockwise order. Then since u must be able to see z, u must be in $\triangle wvp$. Since y, v, u are in clockwise order, u is in $\triangle prv$. Note that requiring u to see z and the fact that $\overline{py}$ does not enter $\mathrm{Vor}(u)$ prevents u from being in $\triangle pyv$, and u is not in $\triangle pry$ by the choice of y. Therefore $u \notin \triangle prv$, and this contradiction concludes the base case.

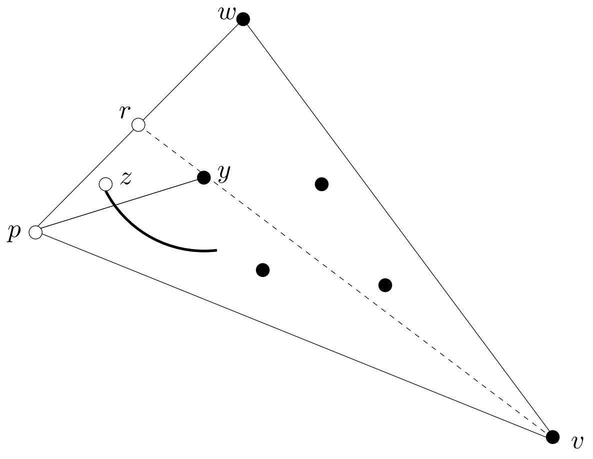

Figure 3.10: z is the Voronoi vertex of v, y, and another unknown site u. $r = \overrightarrow{vy} \cap \overline{pw}$.

Induction: Assume that if $\overline{py}$ intersects $k-1$ Voronoi cells, $y \in \triangle wvp$, and triangles $\triangle pys$ and $\triangle pyr$ contain no sites (where $p \in \text{Vor}(\{w, v\}), s = \overrightarrow{wy} \cap \overline{pv}$, and $r = \overrightarrow{vy} \cap \overline{pw}$ as above), then the Voronoi arcs and vertices cannot all be wedged.

It remains to show that if $\overline{py}$ intersects k cells, the Voronoi arcs and vertices cannot all be wedged. As in the base case, there are two cases here, one where the first cell intersected by $\overline{py}$ is $\text{Vor}(w)$, and the other where the first cell intersected by $\overline{py}$ is $\text{Vor}(v)$.

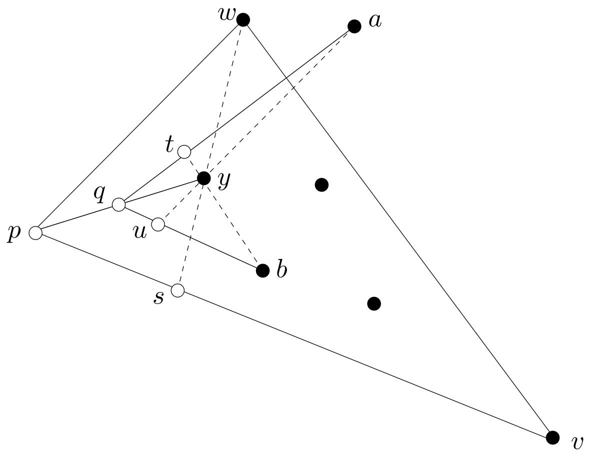

Figure 3.11: Possible locations for sites a and b. $q = \overline{py} \cap \text{Vor}(\{a, b\}), u = \overrightarrow{ay} \cap \overline{qb}, t = \overrightarrow{by} \cap \overline{qa}$.

Case I: Suppose that the first cell intersected by $\overline{py}$ is $\text{Vor}(w)$. The next step of the proof is determined by whether or not $\overline{py}$ intersects a cell owned by a site that is on the opposite side of $\overline{py}$ from w.

If $\overline{py}$ does intersect at least one cell owned by a site that is on the opposite side of $\overline{py}$ from w, let Vor(b) be the first such cell. Let a be the site whose cell is intersected by $\overline{py}$ immediately before b's cell (it may be that $a = w$). Let $q = \overline{py} \cap \text{Vor}(\{a, b\}), u = \overrightarrow{ay} \cap \overline{qb}, t = \overrightarrow{by} \cap \overline{qa}$.

To see that y is in $\triangle qab$, first note that $q \in \triangle wsp$, but there are no sites in the interior of this triangle. The segment $\overline{qa}$ cannot intersect $\overline{pw}$ because q is visible to a. Since a is on the same side of $\overline{py}$ as w, $\overline{qa}$ must intersect $\overline{wy}$. It is possible that a may be colinear with w and y, so that $a \in \overline{wy}$. Likewise, $\overline{qb}$ must intersect $\overline{ys}$. However, b cannot be colinear with y and s, by the choice of y. Therefore, y is strictly inside $\triangle qab$.

Since $\overline{aq}$ intersects $\overline{wy}$, the point u must be in $\triangle pys$. By the choice of y, $\triangle pys$ contains no sites, so $\triangle qyu \subset \triangle pys$ is also empty. Likewise, $t \in \triangle wyp$, and $\triangle wyp$ is empty, so $\triangle qty$ is empty as well.

The segment $\overline{qy}$ passes through at most $k-1$ cells, so the inductive hypothesis may be applied, and the lemma holds.

Otherwise, $\overline{py}$ does not intersect any cell owned by a site that is on the opposite side of $\overline{py}$ from w. Let c be the site that owns the last cell intersected by $\overline{py}$ before Vor(y). Let $u = \overrightarrow{cy} \cap \overline{pv}$. As in the base case, let z_1 be the Voronoi vertex on Vor($\{c, y\}$) on the opposite side of $\overline{py}$ from c. Let d_1 be the site other than c and y that owns z_1 . Then z_1 sees the cells c, y, d_1 in clockwise order.

If the sites c, y, d_1 are not be in clockwise order, z_1 is not wedged and the lemma holds. So suppose that sites c, y, d_1 are in clockwise order. Then d_1 must be in the shaded region in Fig. 3.12 since $\triangle pws$ is empty and $\overline{d_1 z_1}$ may not intersect $\overline{pw}$ or $\overline{pv}$ by the Visibility Theorem.

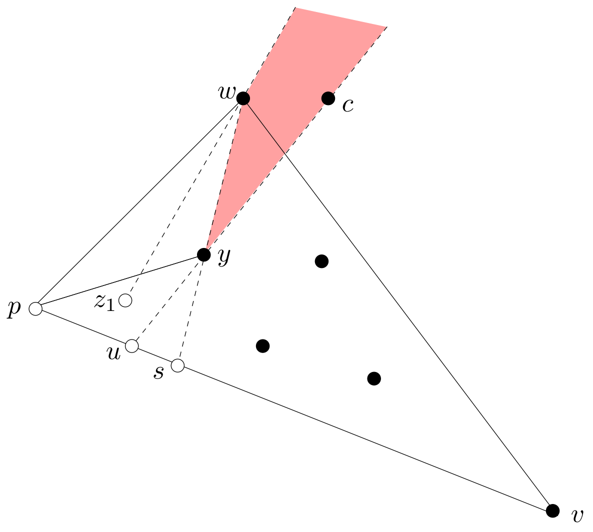

Figure 3.12: c is the site that owns the last cell intersected by $\overline{py}$ before Vor(y). d_1 must lie in the shaded region.

There must be an arc Vor($\{d_1, y\}$) incident to z_1 . Follow this arc away from z_1 . The same reasoning that was used to show the existence of z in the base case shows that $\mathrm{Vor}(\{d_1, y\})$ is not an ellipse and does not leave $\triangle pyu_1$, where $u_1 = \overrightarrow{d_1y} \cap \overline{pv}$. Let z_2 be the Voronoi vertex of $\mathrm{Vor}(\{d_1, y\})$ other than z_1 , and let d_2 be the third owner of z_2 . Then z_2 sees the cells d_1, y, d_2 in clockwise order. If the sites d_1, y, d_2 are in counterclockwise order or are colinear, then z_2 is not wedged and we are done. So suppose d_1, y, d_2 are in clockwise order. By repeating the above argument, we can generate sequences $z_1, z_2, \ldots$ and $d_1, d_2, \ldots$, where $z_i = \mathrm{Vor}(\{d_i, y, d_{i+1}\})$, z_i is visible to d_i, y, d_{i+1}, z_i sees the cells d_i, y, d_{i+1} in clockwise order, and the sites d_i, y, d_{i+1} are in clockwise order.

Note that each site d_i lies between $\overrightarrow{yw}$ and $\overrightarrow{yd_{i1}}$. Therefore, each site in S appears at most once in the sequence $d_1, d_2, \ldots$. The set S is finite, so the sequences will eventually terminate, say at z_N and d_N. Let f be the site other than d_N and y that owns z_N . Then either d_N, y, f are not in clockwise order or f cannot see z_N, because otherwise the sequences would not have stopped at z_N and d_N. In either case, z_N is not wedged and the lemma holds.

Case II: Suppose, on the other hand, that the first cell intersected by $\overline{py}$ is $\mathrm{Vor}(v)$.

If $\overline{py}$ intersects at least one cell owned by a site on the opposite side of $\overline{py}$ from v, let $\mathrm{Vor}(a)$ be the first such cell. Let b be the site whose cell is intersected by $\overline{py}$ immediately before a's cell (it may be that $b = v$). Then the same reasoning as in Case I can be used.

Otherwise, $\overline{py}$ does not intersect any cell owned by a site on the opposite side of $\overline{py}$ from v. The argument for the analogous situation from Case I can be applied with the modification that d_i, y, d_{i+1} are counterclockwise. □

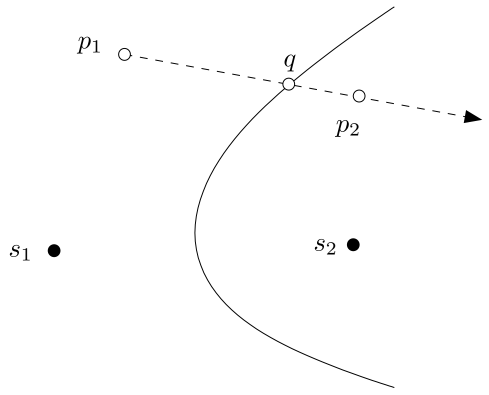

Figure 3.13: All of the points on the ray that are beyond q are dominated by s_2 over s_1.

Lemma 6 *In any anisotropic Voronoi diagram including sites a, b, and y for which $y \in \triangle abp$ for some point p on* $\mathrm{Vor}(\{a, b\})$ *the Voronoi arcs and vertices cannot all be wedged.*

Proof: Let u be the site whose Voronoi cell is first entered by ray $\overrightarrow{py}$. If $u = a$ or $u = b$, the result follows from lemma 5. Otherwise, u is distinct from a and b. Assume that a, b, p are in clockwise order. Then p must see the cells a, u, b in clockwise order. If the sites a, u, b are in counterclockwise order or are colinear, then p is not wedged and the lemma holds. So assume that the sites a, u, b are in clockwise order. Then u and p are on opposite sides of the line through a and b, and u is outside of $\triangle abp$. The site u cannot be on $\overrightarrow{py}$ since either y would block u's view of p and p would not be wedged, or u would be on $\overline{py}$, contradicting the fact that u is outside of $\triangle abp$.

The site u must either be on the same side of the ray $\overrightarrow{py}$ as a or on the same side of the ray as b. Suppose u is on the same side of $\overrightarrow{py}$ as a. Then $y \in \triangle ubp$ by the above reasoning about the position of u. In this case, apply lemma 5 with $w = u$ and $v = b$. Otherwise, u is on the same side of $\overrightarrow{py}$ as b and $y \in \triangle aup$, so the result follows from lemma 5 with $w = u$ and $v = a$. □

Fact 7 *Let s_1 and s_2 be sites, and p_1, p_2 points in the plane. Suppose that s_1 dominates p_1 over s_2 . If there is a point q on the ray $\overrightarrow{p_1p_2}$ that s_2 dominates over s_1, then s_2 also dominates over s_1 all of the points on $\overrightarrow{p_1p_2}$ that are beyond q. In other words, once a ray passes into s_2's territory, it never returns to s_1's. See Fig. 3.13.*

Fact 8 *Let s_1, s_2 be sites and let p_1, p_2 be points such that s_1 dominates p_1 over s_2, and s_2 dominates p_2 over s_1. A direct consequence of Fact 7 is that $\overline{s_1p_1}$ cannot cross $\overline{s_2p_2}$.*

Lemma 9 *After any iteration of the incremental insertion algorithm, let $s_1, s_2, \ldots, s_n$ be the sites of v's star, labeled so that $s_1, s_2, \ldots, s_n$ are in clockwise order around v. Let p_i be a point on* $\mathrm{Vor}(\{v, s_i\})$ *that v and s_i dominate over s_i's neighbors in v's current star (since s_i is currently in v's star, p_i must exist). Then $p_1, p_2, \ldots, p_n$ must also be in clockwise order around v.*

Proof: Base case: $n = 3$. Fix the locations of s_1, s_2, s_3, p_1, and p_3 anywhere in the plane such that s_1, s_2, s_3 are clockwise around v, and $s_1, s_2 \notin \triangle vs_3p_3$, and $s_2, s_3 \notin \triangle vs_1p_1$. Since s_1 and v dominate p_1 over s_2 and since s_3 and v dominate p_3 over s_2, the path $P = s_1, p_1, v, p_3, s_3$ forms a barrier that cannot be crossed by $\overline{s_2p_2}$ or $\overline{vp_2}$, by Fact 8. See Fig. 3.14.

Let R_1 be the region between and including $\overrightarrow{vs_1}$ and $\overrightarrow{vs_3}$, sweeping counterclockwise around v from $\overrightarrow{vs_1}$. Note that s_1, s_2, s_3 appear in clockwise order around v, so that R_1 is on the opposite side of P from s_2. Let q be any point in R_1. Then either q is not in the allowable region for s_2 and v, or $\overline{s_2q}$ intersects P. Therefore, p_2 cannot be in R_1.

If $\overrightarrow{vp_1} \notin R_1$, let R_2 be the region between and including $\overrightarrow{vp_1}$ and $\overrightarrow{vs_1}$, sweeping counterclockwise around v from $\overrightarrow{vp_1}$. Let $q \in R_2$. Since $s_2 \notin \triangle vs_1p_1$, either $\overline{s_2q}$ intersects $\overline{vp_1}$ or $\overline{s_1p_1}$, or $\overline{vq}$ intersects $\overline{s_1p_1}$. By Fact 8 none of these crossings are allowed, so $p_2 \notin R_2$.

Likewise, if $\overrightarrow{vp_3} \notin R_1$, let R_3 be the region between and including $\overrightarrow{vs_3}$ and $\overrightarrow{vp_3}$, sweeping clockwise around v from $\overrightarrow{vp_3}$. By the same reasoning as above, $p_2 \notin R_3$.

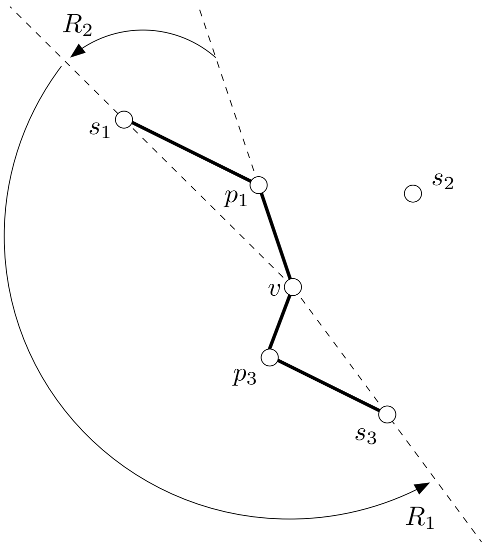

Figure 3.14: Examples of the regions R_1 and R_2 in lemma 9.

So, the only portion of the plane the may contain p_2 lies in the region between $\overrightarrow{vp1}$ and $\overrightarrow{vp3}$, sweeping clockwise around v from $\overrightarrow{vp_1}$. In other words, p_1, p_2, p_3 must be in clockwise order around v.

Induction: Suppose that if v's star contains $n - 1$ sites s_i in clockwise order around v, the corresponding points p_i must also be in clockwise order around v.

If v's star contains n sites, temporarily remove a site, say s_2 from the star. Then for each remaining site, choose a point $p_i \in \text{Vor}(\{v, s_i\})$ that s_i dominates over its neighbors. For s_1, choose p_1 such that s_1 also dominates p_1 over s_2. For s_3, choose p_3 such that s_3 also dominates p_3 over s_2. Since the s_i's are able to coexist in v's star, such p_i's must all exist. By the inductive hypothesis, these $n - 1$ points p_i are in clockwise order around v.

Now consider the star of v that contains only s_1, s_2, and s_3. Let p_1 and p_3 be the same points as above. Find a point p_2 that s_2 dominates over s_1 and s_3. Again by the inductive hypothesis, p_1, p_2, p_3 are in clockwise order around v. Therefore, in the star with all n sites, all n of the p_i's are also in clockwise order around v, as desired. $\square$

Theorem 10 (Equivalence Theorem) *If the set of sites S has been refined enough so that all the arcs and vertices of the anisotropic Voronoi diagram are wedged and the dual of the anisotropic Voronoi diagram is a triangulation, then the star of a site v produced by the incremental insertion* *algorithm contains the same sites as* star(v) *in the dual of the anisotropic Voronoi diagram of* S*.*

Proof: The proof proceeds by first showing that all of the sites that are in star(v) in the dual of the anisotropic Voronoi diagram are inserted into the star of v created by the incremental insertion algorithm and are never removed from it. Then it is shown that all of the sites which do not appear in star(v) in the dual of the anisotropic Voronoi diagram also do not appear in the star output by the algorithm.

Let x be a site in star(v) in the dual of the anisotropic Voronoi diagram of S. Then the arc $\mathrm{Vor}(\{v, x\})$ appears in the anisotropic Voronoi diagram of S. Since all the Voronoi arcs are wedged, this arc is visible to both x and v. Let p be a point on the arc. Since $p \in \mathrm{Vor}(x)$ and p is visible to x in the anisotropic Voronoi diagram of S, p is also in $\mathrm{Vor}(x)$ and is visible to x in any 2-site anisotropic Voronoi diagram containing x. So x dominates p over any other site in S (except v). Likewise, v dominates p over any other site in S (except x). Because all of the Voronoi arcs and vertices are wedged, p must be in the allowable region of v and x by lemma 6. Therefore, the incremental insertion algorithm will always insert and never remove a site that belongs in v's star.

It remains to show that any site not in star(v) in the dual of the anisotropic Voronoi diagram will not be in the star produced by the algorithm.

Let $\triangle vxy$ be a triangle in star(v) in the dual of the anisotropic Voronoi diagram of S. Then x and y must each be inserted into v's star at some point during the algorithm, and once they have been inserted, neither of them will be removed. Suppose that x is inserted into v's star before y, and that sites $w_1, w_2, \ldots, w_n$ which subtend $\triangle vxy$ are also inserted before y. Further suppose that the sites are oriented as in Fig. 3.15. Assume that when y is inserted into v's star, the sites w_i are not removed from the star and seek a contradiction.

Since the sites $y, w_1, w_2, \ldots, w_n, x$ coexist in v's star by assumption, there must be points $p_y, p_1, p_2, \ldots,\ p_n, p_x$ such that $p_y \in \mathrm{Vor}(\{v, y\})$, $p_i \in \mathrm{Vor}(\{v, w_i\})$, $p_x \in \mathrm{Vor}(\{v, x\})$, and each site dominates the corresponding p over its neighbors. Since $y, w_1, w_2, \ldots, w_n, x$ are in clockwise order around v, the points $p_y, p_1, p_2, \ldots, p_n, p_x$ must also be in clockwise order around v, for *any* choices of the p's that satisfy the above criteria, by lemma 9. Let $q = \mathrm{Vor}(\{v, x, y\})$, the Voronoi vertex of v, x, and y, and let $p_y = p_x = q$. Then $p_i, 1 \le i \le n$, must lie between $\overrightarrow{vq}$ and $\overrightarrow{vq}$ that is, on $\overrightarrow{vq}$. However, w_i cannot dominate any points on $\overline{vq}$ over x or y, and v cannot dominate any points on $\overrightarrow{vq}$ beyond q over x or y. Thus, the p_i do not exist and the sites $w_1, w_2, \ldots, w_n$ cannot coexist in v's star with x and y, the desired contradiction. The incremental insertion algorithm will therefore remove each w_i, beginning with w_1.

The incremental insertion algorithm inserts all of the sites that are in star(v) in the dual of the

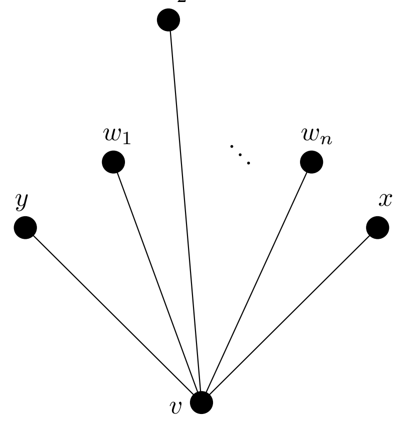

Figure 3.15: The portion of v's star between y and x, just after y has been inserted, but before checking if y's neighbors in the star should remain

anisotropic Voronoi diagram and removes or does not insert the sites that are not in star(v), so it produces the desired output.

□

Chapter 4

Refinement

We saw in the previous chapter that given a set V of sites, the independently constructed stars are guaranteed to agree with each other if all of the arcs and vertices of the anisotropic Voronoi diagram of V are wedged. In most cases, the anisotropic Voronoi diagram of the input sites will not satisfy this criterion. We must therefore *refine* by adding additional sites. Refinement will also ensure that the triangles in each of the stars will be of high quality. Once all of the stars agree with each other and the triangles in the stars are of suitable quality, the final mesh is obtained simply by combining the stars into a single triangulation.

The goal of a refinement algorithm is to find a point p in the domain at which a new site may be inserted. The point p should not be too close to any existing sites, because if it were, the refinement algorithm might continue refining forever and never terminate.

Despite much effort, we were unable to devise a provably reliable algorithm for refinement. We did however, find two heuristics that performed satisfactorily in practice, one using Voronoi arcs to locate the positions of new sites and the other using Poisson darts.

4.1 Voronoi Arc Heuristic

The mesh generation algorithm begins by constructing the stars of the input sites. If any of the stars do not agree with each other or contain poor quality elements, refinement begins.

First, encroachments are eliminated.

Definition 7 *In the work of Labelle and Shewchuk [17], an input segment s (or a subsegment of an input segment) is said to be* encroached *by a site w if* $\mathrm{Vor}(w)$ *intersects s and w is not an endpoint of s.*

Definition 8 *Here, segment $s = \overline{ab}$ is defined to be* encroached *by w if $w \neq a, b$ and w's Voronoi cell intersects s in the 3-site anisotropic Voronoi diagram of a, b, and w.*

The second definition is simpler to implement with the star-based algorithm, since $\mathrm{Vor}(w)$ is unknown. Also, a segment s is encroached according to the first definition if and only if it is encroached according to the second definition. To see why, suppose $s = \overline{ab}$ is encroached by a site w according to the first definition. Then $\mathrm{Vor}(w)$ must intersect s in the 3-site anisotropic Voronoi diagram of a, b, and w, and so s is also encroached by the second definition. If w encroaches s by the second definition, then $\mathrm{Vor}(w) \cap s$ is not owned by the endpoints of s in the true anisotropic Voronoi diagram, so s is also encroached by the first definition.

If segment $s = \overline{ab}$ is encroached, we split s by inserting a new site at its "midpoint" $m = \mathrm{Vor}(\{a, b\}) \cap s$. The resulting subsegments are split repeatedly until they are no longer encroached.

New sites are inserted as long as there are "bad" triangles. A triangle is "bad" if it is *disagreeing* or contains an angle smaller than a user-supplied angle bound, when measured from the perspective of any of the triangle's vertices. Triangles with small angles are also referred to as *poor quality* triangles.

Definition 9 *Let $t = \triangle uxw$ be a triangle in u's star. Then t is a* disagreeing triangle *if any of the following hold:*

- *edge $\overline{ux}$ does not appear in x's star*
- *edge $\overline{uw}$ does not appear in w's star*
- *$\triangle uxw$ does not appear in x's star*
- *$\triangle uxw$ does not appear in w's star*

Labelle and Shewchuk insert new sites at Voronoi vertices that dualize to poor quality triangles and on unwedged portions of Voronoi arcs. In the star-based algorithm, the true Voronoi arcs are not known, but this approach inspired our heuristic.

Given a bad triangle, the heuristic considers the 3-site anisotropic Voronoi diagram of the triangle's vertices. It will insert a site at a Voronoi vertex in this diagram if the new site would be far enough away from *all* of the existing sites (not just the 3 sites in the diagram) and would be in the domain. Otherwise, we insert at a point where one of the Voronoi arcs in the 3-site diagram leaves its wedge, if that point is far enough away from all of the other sites and lies in the domain. If neither of these approaches succeeds, the heuristic moves on to the next bad triangle. Also, new sites are not permitted to encroach upon any subsegment. If a new site would encroach on a subsegment, it is not inserted; instead the subsegment is split at its "midpoint."

Formally, let $t = \triangle uwx$ be a bad triangle, V be the set of all the sites, and Ω be the input domain. Then REFINE(t, V, Ω) will find a location where a new site may be inserted in an attempt to eliminate t.

```
REFINE($t, V, \Omega$) :
    $v_1, \ldots, v_i$ ($i \leq 4$) $\leftarrow$ Voronoi vertices in the 3-site $\{u, w, x\}$ anisotropic Voronoi diagram
    for $k \leftarrow 1$ to $i$
        if $v_k \in \Omega$ and distance from all existing sites to $v_k > tol$
            return $v_k$
    $p_1, \ldots, p_j$ ($j \leq 6$) $\leftarrow$ points where Voronoi arcs leave their wedges in the 3-site diagram
    for $k \leftarrow 1$ to $j$
        if $p_k \in \Omega$ and distance from all existing sites to $p_k > tol$
            return $p_k$
    return FAILEDTOFINDNEWSITE
```

tol is a user-defined parameter that controls how closely the sites may be spaced. Note that it is possible that a location for a new site is not found. When angle bounds of 15° or smaller were requested, this was not a problem in our experiments; see chapter 5. Labelle and Shewchuk proved that their algorithm terminates for angle bounds of up to 20°, and so one would expect this heuristic to perform similarly well in practice.

If a graded mesh is desired, the user would provide a routine that could be queried to see if a given triangle is too large. Then the definition of a "bad" triangle can be extended to include triangles that are too large. This extension will cause overly large triangles to be eliminated through refinement.

4.2 Poisson Darts

The Poisson dart heuristic is quite similar to the Voronoi arc heuristic. The only difference between these methods is the REFINE routine. Rather than use the structure of the stars to decide where to insert a new site, "darts" are randomly thrown into the domain as long as there are disagreeing or poor quality triangles. A dart is accepted as the location of a new site if, for each existing site, the distance between the site and the dart is large enough as measured by both the dart and the existing site. If a dart would encroach upon a segment, the segment is split as above unless splitting the segment would result in a too-short segment, in which case the dart is discarded and the segment is not split. If too many darts are rejected in a row, then the definition of "too close" is changed and sites are allowed to be closer together.

Let V be the current set of sites and Ω the domain to be meshed.

```
Refine($V, \Omega$) :
    $rejected \leftarrow 0$
    while True
        if ($rejected \leftarrow max\_rejected$)
            $tol = 0.85\ tol$
            $rejected \leftarrow 0$
        $p \leftarrow$ random point in $\Omega$
        if distance from all existing sites to $p > tol$
            return $p$
        else
            $rejected$++
```

max_rejected is a user-defined parameter that controls how many sites in a row are rejected before *tol* is reduced. In our experiments, we found that $max\_rejected = 100$ worked well; see chapter 5. One possible optimization to this heuristic is, given a bad triangle t, to throw darts only in the vicinity of t, rather than at the entire domain. This optimization was used in the experiments.

Chapter 5

Results

In order to test the heuristics of the previous chapter, we generated a variety of triangulations. The input to each experiment was a square domain with ten random points in its interior and an angle bound. In practice, we found bounds of 20° and 10° to work well for the Voronoi arc and Poisson dart heuristics, respectively. It is important to note that for any given angle bound, we have not proven that refinement will terminate.

The *swirl* and *sink* metric tensors we used are described by François Labelle [1]; see Fig. 5.1 and Fig. 5.2 for visualizations of these metric tensor fields.

5.1 Voronoi Arc Heuristic

The *swirl* and *sink* metric tensor fields were used to generate meshes with a variety of angle bounds, 5°, 10°, 15°, and 20°. See Fig. 5.3 for the triangulations that used the swirl tensor and Fig. 5.4 for the triangulations that used the sink tensor.

In the *swirl* triangulations, the triangles near the center of the domain are fairly round, while those near the edges of the square are much more stretched. The shapes of the triangles agrees with the shapes indicated by the metric tensor. The *sink* triangulations exhibit similar behavior, isotropic triangles near the center and triangles that are stretched according to the metric in the rest of the domain.

As expected, more refinement is needed when a larger angle bound is requested for both metric tensor fields.

[1] http://www.eecs.berkeley.edu/ flab/cs294-5/project2/mesh.html

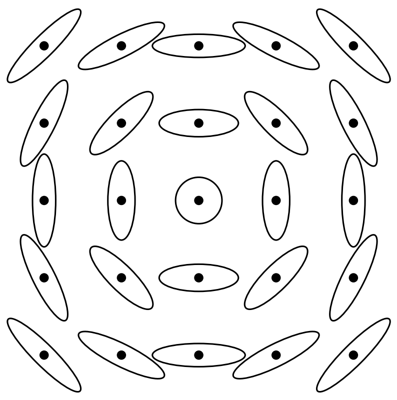

Figure 5.1: The *swirl* metric tensor field, with isocontours shown at select points

.

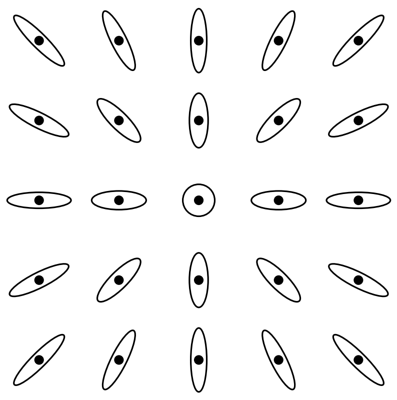

Figure 5.2: The *sink* metric tensor field, with isocontours shown at select points

.

5.2 Poisson Darts

The same metric tensor fields were used with the Poisson darts heuristic. This refinement method was much slower than the Voronoi arcs method, because many of that points were considered for insertion were ultimately discarded.

The *swirl* triangulations (Fig. 5.5) and the *sink* triangulations (Fig. 5.6) show the same general anisotropy as when the Voronoi arc heuristic was used. The *sink* examples illustrate that with the Poisson darts method, restricting the locations of the darts to be near disagreeing or poor quality triangles clearly restricts the refinement to those parts of the mesh as well. Notice that the top of the triangulation on the left side of the figure and the right of the triangulation on the right side of the figure are sparser than the rest of their respective triangulations because less refinement was needed in those regions. In contrast, a Voronoi vertex that dualizes to a poor quality triangle may

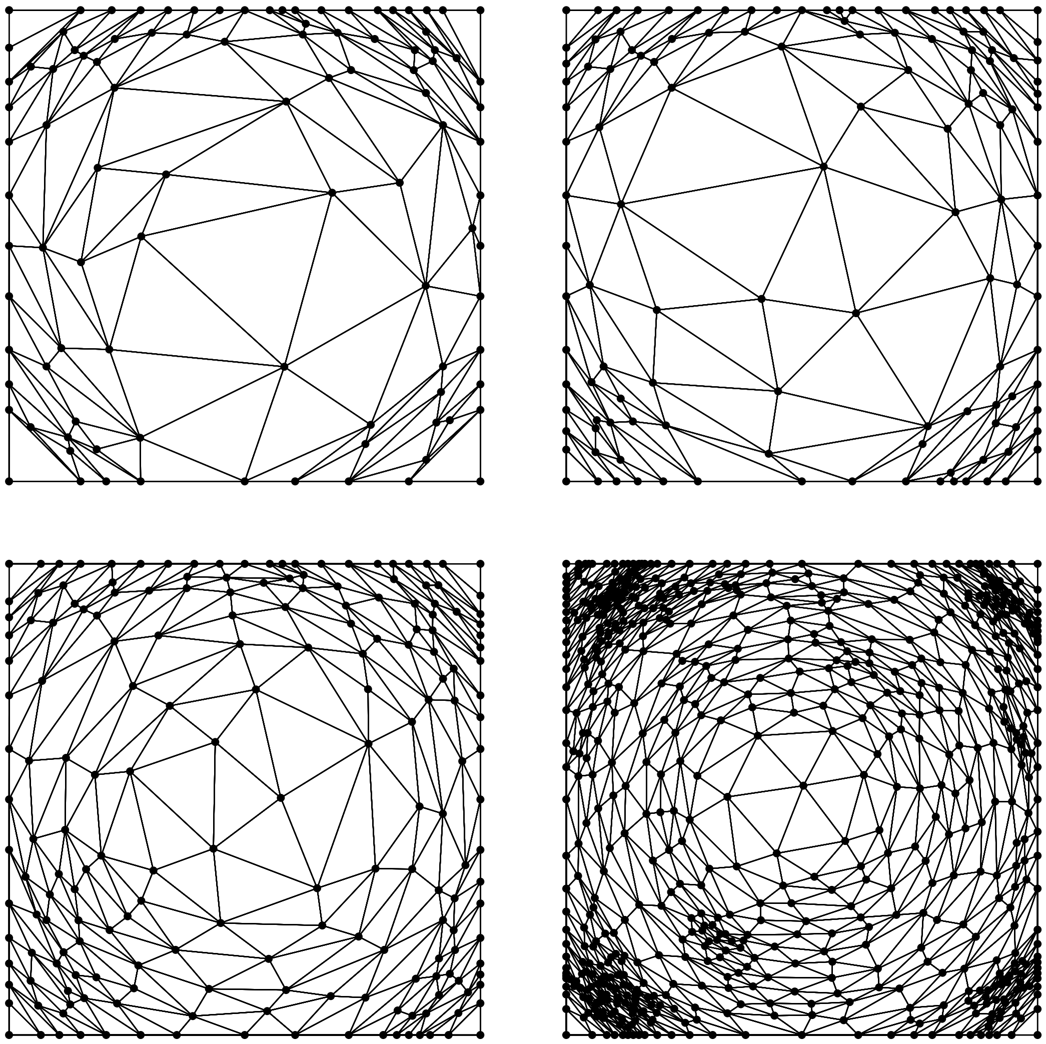

Figure 5.3: Triangulations generated from the *swirl* metric tensor and the Voronoi arc heuristic. Angle bounds: Upper left: 5°; upper right: 10°; lower left: 15°; lower right: 20°.

be far away from the triangle itself.

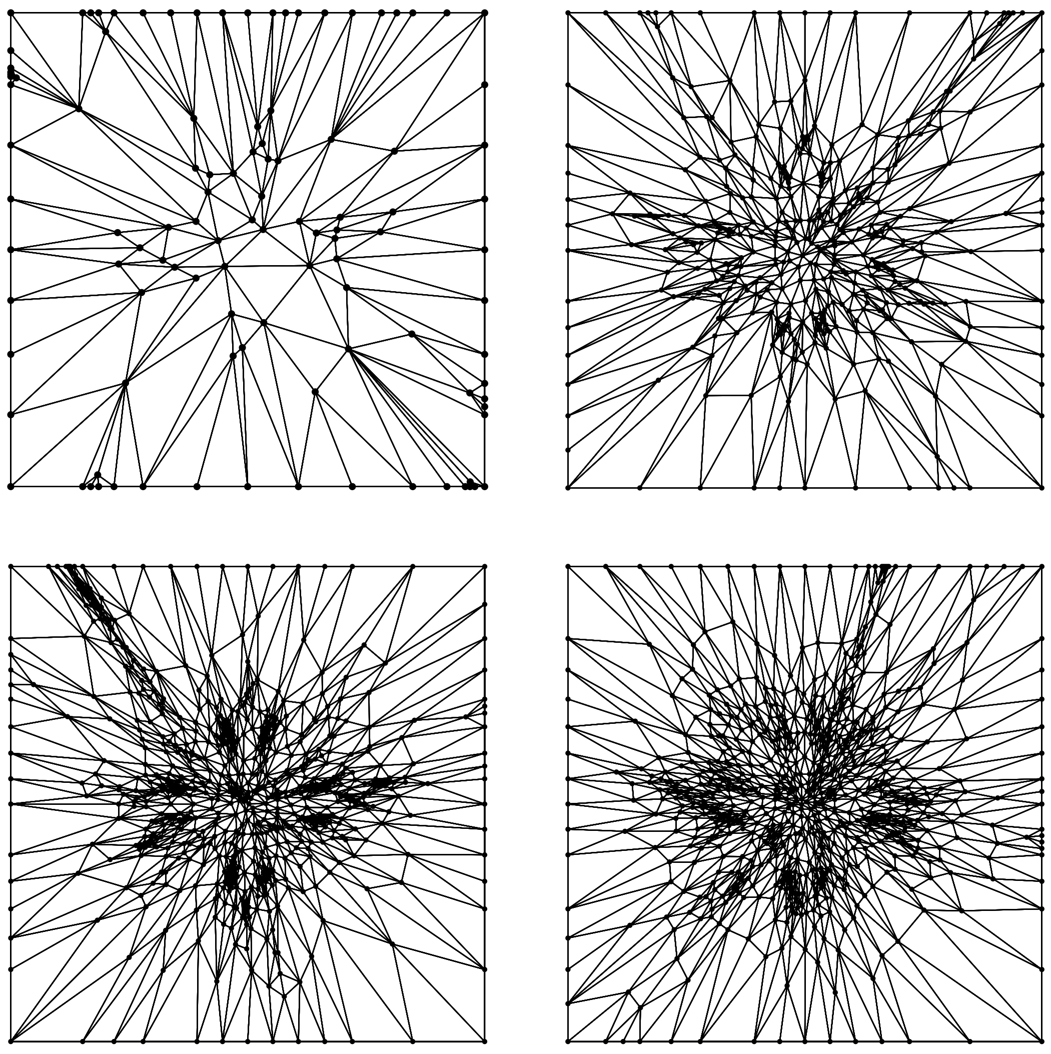
Figure 5.4: Triangulations generated from the *sink* metric tensor and the Voronoi arc heuristic. Angle bounds: Upper left: 5 °; upper right: 10 °; lower left: 15 °; lower right: 20 °.

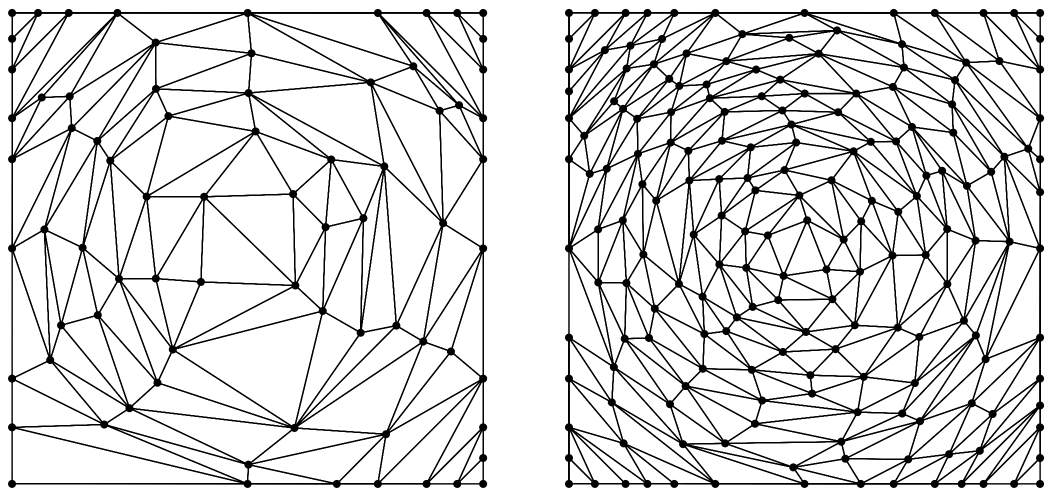

Figure 5.5: Triangulations using the *swirl* metric tensor and the Poisson dart heuristic. Angle bounds: left: $5\,^\circ$, right: $10\,^\circ$.

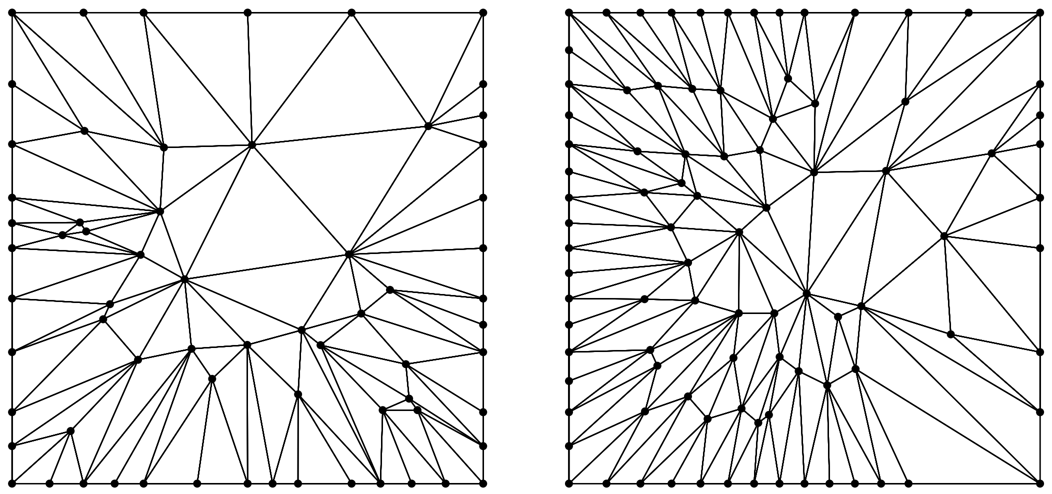

Figure 5.6: Triangulations using the *sink* metric tensor and the Poisson dart heuristic. Angle bounds: left: $5\,^\circ$, right: $10\,^\circ$.

Chapter 6

Future Work

Although the heuristics for refinement presented in chapter 4 perform acceptably in practice, a provably reliable refinement algorithm would be preferable. Finding such an algorithm has proven to be challenging. Another logical step would be to extend the current work to anisotropically mesh surfaces embedded in three dimensions. Anisotropic surface meshing is especially useful for computer graphics.

Appendix A

Notes on Implementing Flips

Let $\triangle vxy$ be a triangle in v's star and let w be a site that subtends $\triangle vxy$. We must decide if w should be inserted into v's star.

1. Let $\mathcal{C}$ be the portion of $\mathrm{Vor}(\{v, w\})$ that is visible to both v and w in the 2-site v, w anisotropic Voronoi diagram.

2. Find the sections of $\mathcal{C}$ that w dominates over y:

   (a) Let $\mathcal{B}_{wy}$ be the boundary that separates points dominated by w from those dominated by y.

   (b) Compute $\mathcal{C} \cap \mathcal{B}_{wy}$. Call the intersection points (if any) $p_1, p_2, \ldots$.

   (c) Sort the p_i's along $\mathcal{C}$. The p_i's divide $\mathcal{C}$ into sections $\mathcal{C}_1, \mathcal{C}_2, \ldots$ alternatingly dominated by w and y.

   (d) Determine which $\mathcal{C}_i$'s are dominated by w and which by y. This test can be accomplished by selecting a point p on $\mathcal{C}_i$ and if $|\overline{wp} \cap \mathrm{Vor}(\{w, y\})| < |\overline{yp} \cap \mathrm{Vor}(\{w, y\})|$, then w dominates p over y.

3. For each $\mathcal{C}_i$ that w dominates over y, find the sections of it that v also dominates over y by repeating step 2 with v in place of w.

4. For each section of $\mathcal{C}$ that w and v dominate over y, find the subsections that w and v also dominate over x by repeating steps 2 and 3 with x in place of y.

5. If any portion of the original $\mathcal{C}$ is dominated by both w and v over both y and x, then insert w into v's star. Otherwise, do not insert w.

Bibliography

[1] Pierre Alliez, David Cohen-Steiner, Olivier Devillers, Bruno Lévy, and Mathieu Desbrun, *Anisotropic Polygonal Remeshing*, ACM Transactions on Graphics, 22 (2003), pp. 485–493.

[2] Thomas Apel, *Anisotropic Finite Elements: Local Estimates and Applications*, 1999.

[3] Jean-Daniel Boissonnat, Camille Wormser, and Mariette Yvinec, *Anisotropic Diagrams: Labelle Shewchuk Approach Revisited*, in Seventeenth Canadian Conference on Computational Geometry, 2005, pp. 266–269.

[4] Houman Borouchaki, Pascal J. Frey, and Paul-Louis George, *Unstructured Triangular-Quadrilateral Mesh Generation. Application to Surface Meshing*, in Fifth International Meshing Roundtable, October 1996, pp. 229–242.

[5] Houman Borouchaki, Paul-Louis George, Frédéric Hecht, Patrick Laug, and Eric Saltel, *Delaunay Mesh Generation Governed by Metric Specifications. Part I. Algorithms*, Finite Elements in Analysis and Design, 25 (1997), pp. 61–83.

[6] Frank J. Bossen and Paul S. Heckbert, *A Pliant Method for Anisotropic Mesh Generation*, in Fifth International Meshing Roundtable, October 1996, pp. 63–74.

[7] Adrian Bowyer, *Computing Dirichlet Tessellations*, Computer Journal, 24 (1981), pp. 162–166.

[8] Gustavo C. Buscaglia and Enzo A. Dari, *Anisotropic Mesh Optimization and its Application in Adaptivity*, International Journal for Numerical Methods in Engineering, 40 (1997), pp. 4119–4136.

[9] Siu-Wing Cheng, Tamal K. Dey, Edgar A. Ramos, and Rephael Wenger, *Anisotropic Surface Meshing*, in Seventeenth Annual Symposium on Discrete Algorithms, New York, NY, USA, 2006, ACM, pp. 202–211.

[10] O. Courchesne, François Guibault, Julien Dompierre, and Farida Cheriet, *Adaptive Mesh Generation of MRI Images for 3D Reconstruction of Human Trunk*, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2007, pp. 1040–1051.

[11] Eduardo F. D'Azevedo, *Optimal Triangular Mesh Generation by Coordinate Transformation*, SIAM Journal on Scientific and Statistical Computing, 12 (1991), pp. 755–786.

[12] Eduardo F. D'Azevedo and R. Bruce Simpson, *On Optimal Triangular Meshes for Minimizing the Gradient Error*, Numerische Mathematik, 59 (1991), pp. 321–348.

[13] Pascal J. Frey and Frédéric Alauzet, *Anisotropic Mesh Adaptation for Transient Flows Simulations*, in Twelfth International Meshing Roundtable, 2003, pp. 335–348.

[14] Paul-Louis George and Houman Borouchaki, *Delaunay Triangulation and Meshing: Application to Finite Elements*, Hermès, Paris, 1998.

[15] Cyril Gruau and Thierry Coupez, *3D Tetrahedral, Unstructured and Anisotropic Mesh Generation with Adaptation to Natural and Multidomain Metric*, Computer Methods in Applied Mechanics and Engineering, 194 (2005), pp. 4951–4976.

[16] Robert Haimes and Michael J. Aftosmis, *Watertight Anisotropic Surface Meshing Using Quadrilateral Patches*, in Thirteenth International Meshing Roundtable, 2004, pp. 311–322.

[17] François Labelle and Jonathan Richard Shewchuk, *Anisotropic Voronoi Diagrams and Guaranteed-Quality Anisotropic Mesh Generation*, in Nineteenth Annual Symposium on Computational Geometry, New York, NY, USA, 2003, ACM, pp. 191–200.

[18] Xiangrong Li, Jean-François Remacle, Nicolas Chevaugeon, and Mark S. Shephard, *Anisotropic Mesh Gradation Control*, in Thirteenth International Meshing Roundtable, September 2004, pp. 401–412.

[19] Xiang-Yang Li, Shang-Hua Teng, and Alper Üngör, *Biting Ellipses to Generate Anisotropic Mesh*, in Eighth International Meshing Roundtable, 1999, pp. 97–108.

[20] Konstantin Lipnikov and Yuri Vassilevski, *Error Estimates for Hessian-Based Mesh Adaptation Algorithms with Control of Adaptivity*, in Thirteenth International Meshing Roundtable, September 2004, pp. 345–352.

[21] H. Lo and X. Wang, *Generation of Anisotropic Mesh by Ellipse Packing over an Unbounded Domain*, Engineering with Computers, 20 (2005), pp. 372–383.

[22] Shmuel Rippa, *Long and Thin Triangles Can Be Good for Linear Interpolation*, SIAM Journal on Numerical Analysis, 29 (1992), pp. 257–270.

[23] Jonathan Richard Shewchuk, *What is a Good Linear Finite Element? Interpolation, Conditioning, Anisotropy, and Quality Measures.* Manuscript in progress, 2002.

[24] Kenji Shimada, Atsushi Yamada, and Takayuki Itoh, *Anisotropic Triangular Meshing of Parametric Surfaces via Close Packing of Ellipsoidal Bubbles*, in Sixth International Meshing Roundtable, 1997, pp. 375–390.

[25] Ko-Foa Tchon, Julien Dompierre, Marie-Gabrielle Vallet, and Ricardo Camarero, *Visualizing Mesh Adaptation Metric Tensors*, in Thirteenth International Meshing Roundtable, September 2004, pp. 353–364.

[26] David F. Watson, *Computing the n-dimensional Delaunay Tesselation with Application to Voronoi Polytopes*, Computer Journal, 24 (1981), pp. 167–172.