COARSE TETRAHEDRAL MESHING FOR INTERACTIVE SIMULATION

by

David Alexander Stuart

A thesis submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science

School of Computing

The University of Utah

May 2013

The University of Utah Graduate School

STATEMENT OF THESIS APPROVAL

The thesis of David Alexander Stuart

has been approved by the following supervisory committee members:

Adam W. Bargteil	, Chair	3/29/2013
		Date Approved
Joshua A. Levine	, Member	
		Date Approved
Cem Yuksel	, Member	4/2/2013
		Date Approved

and by Alan Davis , Chair of

the Department of School of Computing

and by Donna M. White, Interim Dean of the Graduate School.

ABSTRACT

We present a procedure for generating a coarse, high-quality, tetrahedral mesh whose exterior surface encloses and approximates a given triangle mesh. A tetrahedral mesh is useful for computing perturbation of the triangle mesh based on continuum mechanics: perturbation such as plastic flow, fracture, and elastic deformation. The computer graphics community has long used this physics-based simulation to produce animations of objects exhibiting such physical phenomena. Interactive animation applications such as industrial design, medical training, and computer entertainment require meshes that are particularly efficient and robust, and our meshing procedure targets these properties. We begin with a BCC background lattice and sculpt an initial mesh from it whose tetrahedra occupy some of the volume bounded by the triangle mesh. We then refine this initial mesh with an iterative optimization procedure that simultaneously minimizes the distance from the triangle mesh to the surface of the tetrahedral mesh and maximizes the numerical quality of the tetrahedra. Our procedure provides a trade-off among the mesh's quality, resolution, and degree of approximation of the triangle mesh.

To Gina Vasiloff, whose friendship made this work possible.

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Physics-based animation of deforming, flowing, and fracturing materials is an attractive part of computer graphics both because the problems are easy to express mathematically and because the solutions are so visually interesting. As a result graphics researchers have produced a rapidly growing body of technique for computing these animations, and such animations appear increasingly often in film and computer entertainment.

Computing physics-based perturbation of a given polygonal surface mesh often demands a polyhedral volume mesh on which to compute forces. The finite element method is one such instance. The forces are used to compute the displacement of the volume mesh vertices. One can thence compute the displacement of the surface mesh vertices by embedding the surface mesh in the volume mesh.

The choice of volume mesh can significantly influence the resulting animation in a number of ways. First, the degree to which the volume mesh's surface approximates the original surface mesh (see Figure 1.1) determines how accurately the surface mesh is perturbed when it is embedded in the volume mesh. It also determines how accurately a collision between surface meshes can be approximated by a collision between their respective volume meshes. The surface of the volume mesh must completely enclose the surface mesh (as shown in Figure 1.2): if any part of the surface mesh is exterior to the volume mesh, forces on the volume mesh will not accurately inform the perturbation of

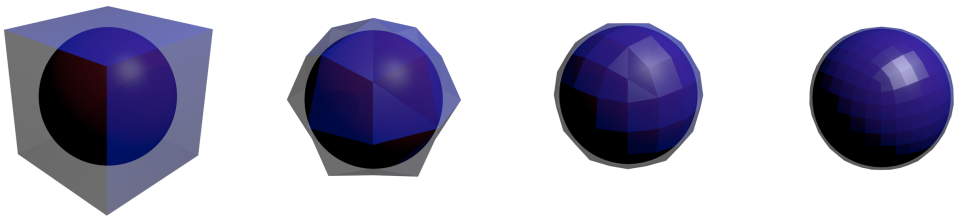

Figure 1.1. A spherical surface mesh is embedded in four different volume meshes. From left to right, each volume mesh's surface approximates the original surface mesh more accurately than that of the previous volume mesh.

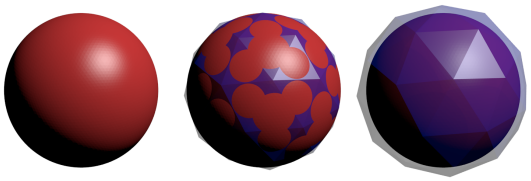

Figure 1.2. A spherical surface mesh (left) is superposed with a nonenclosing (middle) and an enclosing (right) volume mesh.

the surface mesh. Second, the number of elements in the volume mesh determines the computational complexity of computing forces on the whole mesh. A mesh with many elements requires more computational expense than one with few elements. Third, the numerical quality of the elements determines how accurate the computed forces are and, in turn, how stable the simulation is.

Therefore, it is desirable to have a systematic way of generating a volume mesh appropriate for a given simulation and a given surface mesh. There are many such systems available. However, none of them to date produce enclosing volume meshes that effectively satisfy the stability and efficiency requirements of *interactive* simulations. Their surfaces do not accurately represent the input surface mesh while still enclosing it, they have too many elements, or their elements are of low quality. Consequently these meshes, when deployed in the unpredictable and resource-scarce environments of interactive applications, can ignore significant input features, produce imprecise collision detection, exceed computation budgets, and cause poorly conditioned simulations.

This paper documents a meshing procedure that mitigates these problems and provides a trade-off between them. It takes as input a surface mesh of triangular faces and generates as output a coarse volume mesh of high-quality tetrahedral elements whose surface approximates the input. It accomplishes this by first sculpting a background lattice of tetrahedra to a rough initial approximation of the surface mesh. It then optimizes the surface of the sculpted volume mesh for both element quality and proximity to the surface mesh simultaneously. A user can determine the coarseness of the initial lattice (which directly influences the coarseness of the final volume mesh) as well as the maximum distance between the surface of the final volume mesh and the surface mesh (which directly influences the similarity between the surface mesh and the final volume mesh). Both of these parameters can indirectly but reliably influence the quality of the elements in the final mesh.

CHAPTER 2

RELATED WORK

Tetrahedral tessellation of a given volume is a classic set of problems, and the meshing community has addressed it extensively. Our meshing problem of interest—generating an *enclosing* tetrahedral mesh—has appeared in the literature since Capell and colleagues introduced the embedding of a surface mesh in a volumetric control mesh for deformation [9]. That year the same authors argued [10] that a volume mesh that closely fits the embedded surface mesh allows for more accurate deformations than a regular grid like that of Müller [28].

Since then the embedding technique has repeatedly appeared as a practical method of deforming and fracturing surface meshes [17, 24, 36, 40]. The efficacy of a coarse volume mesh for this technique also became apparent during this time [9, 12, 27, 40]. Research on this technique has reaffirmed the perennial desirability of high-quality elements [12, 24].

High-quality elements are a primary goal of most tetrahedral meshing procedures. There are many measures of element quality, as is evident in Shewchuk's survey, "What Is a Good Linear Element" [34]. Some are in common use among graphics researchers, like the ratio of the element's circumradius to its shortest edge [33], and some have less cachet, such as the condition number of the linear transformation between a unit equilateral tetrahedron and the element in question [15]. Different procedures often aim for quality by different measures, but there are some general approaches that can apply to many measures at once.

For instance, a classical mesh generation approach called "advancing front" begins at the volume boundary and sequentially adds well shaped elements based on local heuristics [3]. Advancing front techniques are straightforward to implement, but they lack any guarantees of element quality [21]. They can produce low-quality elements on the surface at sharp corners [5] or in the interior where two advancing fronts meet. This approach is the basis of the popular "NETGEN" program [30], but modern methods do not employ it.

A second and more popular approach is the use of Delaunay triangulations of points scattered throughout the volume. The error in a function's linear approximation defined piecewise on a Delaunay triangulation is theoretically bounded [3]. This is the motivation for Delaunay-based algorithms like Shewchuk's Delaunay refinement [33], Du and Wang's centroidal Voronoi tessellations [13], Si's "TetGen" program [35], and the CGAL 3D mesh generation package. In one of the earliest examples of Delaunay refinement, Edelsbrunner and colleagues begin with an assumed set of points to triangulate [14]. In contrast, choosing the best set of points is a significant corollary problem, and inserting extra points can improve the refinement process [18, 39]. While in two-dimensional meshes the Delaunay approach ensures a lower bound on element quality as measured by an element's minimum dihedral angle, this is not the case in three-dimensional meshes [38]. The Delaunay approach can produce degenerate elements with extremely poor quality, and modified approaches meant to prevent them have severely weakened bounds on error and quality [3].

A third approach, recently gaining considerable traction in the community, begins with a background lattice of high-quality elements and refines it based on the input domain. This concept appears in generation algorithms like the "red-green" subdivision described by Molino and colleagues [25], the level-set technique of Teran and colleagues [37], "isosurface stuffing" by Labelle and Shewchuk [20], and "lattice cleaving" by Bronson and colleagues [7]. A lattice-based algorithm ensures good shape and placement of interior elements and at least encourages the same traits in the surface elements it refines. However, a lattice alone provides no approximation of the volume boundary.

Fourth, an approach employing octrees appears often in the literature. It finely discretizes the volume bounded by the input surface with a grid. It then covers the volume with an octree, refining it until each leaf is entirely inside or outside the discretization [3]. The leaves are then triangulated. In contrast to a lattice-based method an octree-based method produces meshes that sample different parts of their domains at different resolutions: there are many elements on the boundary and fewer in the interior. It also facilitates remeshing in response to a dynamically changing domain [1]. The octree approach is the foundation of the "QMG" technique given by Mitchell and Vavasis [23], and it appears in older methods like those of Buratynski [8], Perucchio and colleagues [29], and Shephard and colleagues [32].

Finally, an important fifth approach to mesh generation is physics-based vertex op-

timization. This approach iteratively refines a mesh with the same methods used for iteratively solving differential equations in a physics-based simulation—the impetus for generating a mesh in the first place! This often involves defining an "energy" value for a given mesh and iteratively changing the vertex positions to reduce the mesh's energy: various energy definitions appear in the literature, such as Centroidal Voronoi Tesselations [13], Optimal Delaunay Triangulations [3], and Hodge-Optimized Triangulations [26]. Both Molino and colleagues [25] and Teran and colleagues [37] propose similar relaxation procedures using a mass-spring system or the finite-element method.

Like these mesh generation techniques, mesh improvement techniques also value element quality. Freitag and colleagues published a widely cited discussion [16] of improving meshes through smoothing vertices (via Laplacian smoothing or other optimizations targeted to a specific quality measure) and local edge swaps. Topological operations like edge swaps often appear in improvement techniques. They include element subdivision, as Liu proposed [22], as well as more dramatic connectivity changes, as in the simplification method by Cutler and colleagues [12] and Klingner and Shewchuk's improvement program "Stellar" [19].

Procedures for generating volume meshes more specific to the embedding technique for simulation aim not only for high-quality elements, but also an enclosing mesh whose surface approximates the input surface. That is, the surface of the volume mesh is everywhere exterior to the surface mesh. As more general meshing algorithms usually do not guarantee this property, an enclosing volume mesh is often obtained by meshing not the input surface but a different surface that is slightly offset from it everywhere in the normal direction. Shen and colleagues suggest this procedure as an application of their implicit surface generation algorithm [31]. This procedure ensures an erroneous approximation of the input, since the offset surface loses the detail of the original surface.

CHAPTER 3

RESEARCH

3.1 Initial Mesh Generation

We take the approach of beginning with a tetrahedral background lattice. This approach provides generality and a known sampling resolution that can be easily tuned. The initial volume mesh taken from this background lattice must be chosen to match the input surface mesh in a way that does not hinder later refinement: our solution is summarized in Algorithm 1.

Algorithm 1 Initial mesh generation

read input surface mesh S
generate volume mesh (V, T) as BCC lattice
$T^{\text{del}} \leftarrow \varnothing$
$T^{\text{rep}} \leftarrow T$
for all $T_i \in T$ **do**
  if T_i does not intersect S **then**
    $T^{\text{del}} \leftarrow T^{\text{del}} \cup \{T_i\}$
  end if
end for
for all $T_k^{\text{rep}} \subseteq T^{\text{del}}$ s.t T_k^{rep} covers all bomb elements in $T \setminus T^{\text{del}}$ **do**
  if $|T_k^{\text{rep}}| < |T^{\text{rep}}|$ **then**
    $T^{\text{rep}} \leftarrow T_k^{\text{rep}}$
  end if
end for
$T \leftarrow (T \setminus T^{\text{del}}) \cup T^{\text{rep}}$

3.1.1 BCC Lattice

We generate an initial tetrahedral volume mesh, based on a body-centered cubic (BCC) crystal lattice, that tessellates the space of the surface mesh's bounding box. The BCC lattice is known from chemistry and is apparent in many physical crystal structures. It is a set of points in $\mathbb{R}^3$ arranged in two identical cubic grids offset from each other by half a cell-width in all three dimensions: the points in each grid are positioned at the

centers of the other grid's cells. The Delaunay triangulation of these points defines a set of tetrahedra, each with two long edges between points in the same grid and four short edges between points in opposite grids (see Figure 3.1).

This mesh has many desirable properties at once. It is isotropic, as the tetrahedra are identical and aligned equally often in three orthogonal directions. Its elements have high quality, each having two dihedral angles of 90° and four dihedral angles of 60° [20]. Its vertices are an optimal sampling of the volume for representing trivariate functions like deformation forces [2]. Ideally we would like to change this mesh as little as possible to retain these properties. Subdivision and connectivity changes can compromise these properties, so we do not employ such topological techniques.

3.1.2 Sculpting

At this point the volume mesh is still shaped like the bounding box. To better approximate the shape of the surface mesh we remove those elements that do not occupy any of the volume bounded by the surface mesh. We determine whether an element occupies some of the bounded volume using a test for each of three different cases, illustrated in Figure 3.2. Our first test is fast, but it can produce false negatives. We check its negative cases further with a pair of slower tests that, together, never produce false negatives.

We first check whether any of the element's four vertices is inside the bounded volume. If an element's vertex is inside, then some nonzero portion of the element is also inside. In many cases most of the elements have this property, and checking it first allows us to

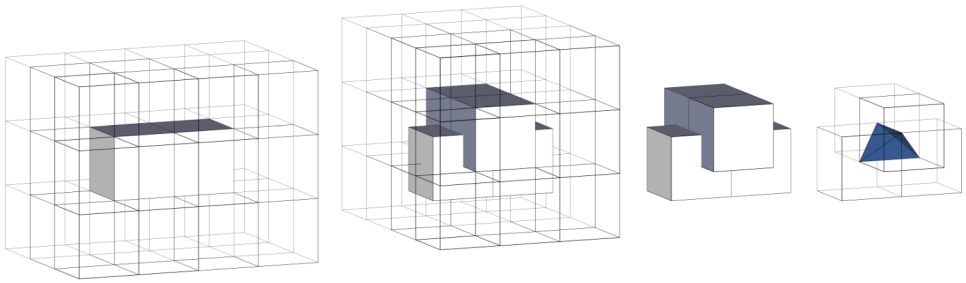

Figure 3.1. The tetrahedra of the BCC lattice. The lattice is constructed from two grids, and the points of each grid lie at the cell centers of the other grid. The tetrahedra are all like the one shown at right: in both grids, two of the tetrahedron's vertices are at the centers of adjacent cells.

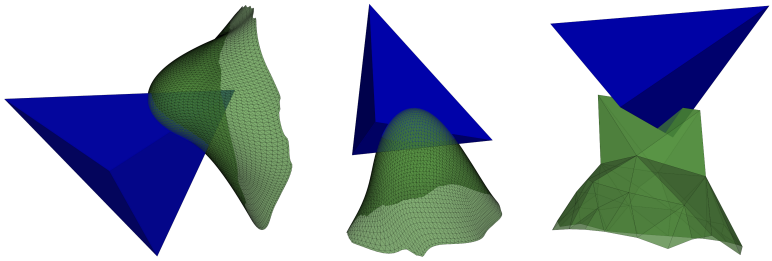

Figure 3.2. Three cases of an element occupying part of the volume bounded by the surface mesh. On the left, a vertex of the element is inside the bounded volume. In the center, a vertex of the surface mesh is inside the element. On the right, an edge of the element intersects two triangles of the surface mesh.

quickly show that those elements occupy some of the bounded volume. We can quickly check a vertex for containment in the bounded volume by evaluating a signed distance field, as described in Section 3.2.3, at the vertex's position.

Other elements may occupy some of the bounded volume without any of their vertices being inside it. Our first test will be negative for these elements. In that case we check for a vertex of the surface mesh being inside the element. Assuming that every surface mesh vertex has at least one triangle incident to it, a surface vertex being inside an element implies that some nonzero portion of the triangle is also inside the element. That means that a portion of the bounded volume close to the triangle intersects that portion of the element's volume. While not as efficient as our first test, this test runs relatively quickly, requiring four dot products for each surface mesh vertex.

If this test is negative, it is still possible that some part of the element is inside the bounded volume. The element can intersect a triangle of the surface mesh in a section that does not contain the triangle's vertices. This can happen even if none of the element's vertices is inside the surface mesh, as Figure 3.2 illustrates: an edge between two vertices outside the surface mesh intersects two surface mesh triangles. To determine if this is the case, we test every edge of the element for intersection with every triangle in the surface mesh.

In truth we only check certain vertices and triangles for these last two tests, as described in Section 3.3.

3.1.3 Defusing the Bombs

After removing all elements that do not occupy any of the volume bounded by the surface mesh there can be some remaining elements with two or more faces on the surface

of the resulting volume mesh—that is, two or more faces not shared with adjacent elements. An example is shown in Figure 3.3. The quality of these two-surface-face elements is ruined in our later optimization. As we push the faces on the surface of the volume mesh to lie close to the surface mesh, these elements' two surface faces spread out, with all four vertices becoming nearly coplanar. A tetrahedron in this shape is called a "sliver," and it has very low quality. We would like our mesh to contain no such "bomb" elements that predictably turn into slivers later.

Definition 1. *Let T be a set of tetrahedra. A* bomb element *is a tetrahedron $T_i \in T$ that has two faces f_0, $f_1 \in T$ such that $\forall\, T_j \in T$, $f_0 \notin T_j$ and $f_1 \notin T_j$.*

As we will show presently the mesh has a bomb element only if the mesh is not the union of sets of 24 elements incident to a common vertex. We call these sets "snowflakes." One is illustrated in Figure 3.4.

Definition 2. *Let $(V_{\mathrm{BCC}}, T_{\mathrm{BCC}})$ be an infinite, tetrahedral BCC lattice tessellating all of $\mathbb{R}^3$. A set $F \subset T_{\mathrm{BCC}}$ is a* snowflake *if $\exists\, v \in V_{\mathrm{BCC}}$ such that $\forall\, T_j \in F$, $v \in T_j$.*

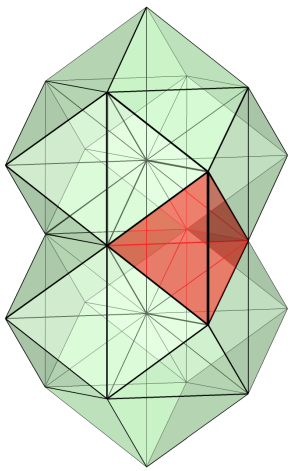

Figure 3.3. An initial mesh with a single bomb element. The bomb element, in front and darkly colored, has two faces on the surface of the mesh. The other elements, lightly colored, each have at most one face on the surface.

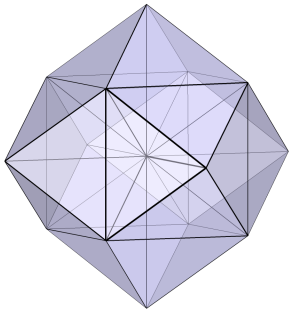

Figure 3.4. A snowflake. Each element in this set is incident to the vertex in the center.

Remark. *No element of a snowflake is a bomb element.*

Proof. Every element T_i in a snowflake is incident to the snowflake's common vertex v, an adjacent body center c, and two vertices a_1 and a_2 that are each adjacent to both v and c. Since there are four choices for a_1 and a_2 another element besides T_i is incident to v, c, and a_1. Therefore the face of T_i defined by those three vertices is shared between two elements. Similarly another element is incident to v, c, and a_2, meaning that a second face of T_i is shared between two elements. Finally, just as a_1 and a_2 are each adjacent to both v and c, there is another body center c' adjacent to v such that a_1 and a_2 are each adjacent to both v and c'. Therefore another element is incident to a_1, a_2, and v, meaning that a third face of T_i is shared between two elements. Hence T_i has at most one unique face and is not a bomb element. □

Proposition. *Let T_{BCC} be the elements of an infinite, tetrahedral BCC lattice tessellating all of $\mathbb{R}^3$. Let S be the set of all snowflakes $F \subset T_{\mathrm{BCC}}$. $\forall\, T \subset T_{\mathrm{BCC}}$, if $\exists\, S_T \subset S$ such that $\bigcup_{F \in S_T} = T$, then T contains no bomb elements.*

Proof. If $\exists\, S_T \subset S$ such that $\bigcup_{F \in S_T} = T$, then $\forall\, T_i \in T$, $\exists\, F \in S_T$ such that $T_i \in F$ and $F \subseteq T$. Therefore every $T_i \in T$ is in a snowflake, all of whose elements are contained in T. Therefore T does not contain a bomb element. □

The mesh is initially a union of snowflakes. If the mesh contains a bomb element after the removals described above, then it is because we removed elements from all snowflakes that contain that element.

We fix this by replacing elements in such a way that every element that was a bomb after the removals is in a complete snowflake after the replacements. For each bomb

element we replace all removed elements that are incident to one of its vertices. This ensures that the snowflake centered at that vertex—one of the snowflakes that contains the bomb element—is part of the mesh.

In choosing which of each bomb element's four vertices to consider for this replacement process we do a global search of the possible combinations of vertices across all bomb elements and choose the one that minimizes the number of replaced elements.

Initially the mesh happens to be a union of snowflakes centered on vertices that all lie in only one of the two staggered grids in the BCC lattice. Because of this we earlier attempted to retain the snowflakes in this particular union that contained an element that intersected the surface mesh. Such a mesh is indeed a union of snowflakes and contains no bomb elements, but it contains far more elements than the mesh described above.

We also attempted to remove bomb elements with a simpler scheme than replacing snowflakes: we subdivided each bomb element into four elements, each incident to the bomb element's centroid. This indeed resulted in a mesh without bomb elements, but the subdivided elements nonetheless became slivers. Intuitively this is because there were no new vertices to update on the surface of the volume mesh. The faces of the original bomb element were still pressed into a sliver even though the sliver was subdivided.

3.2 Mesh Optimization

At this point we have an enclosing volume mesh free of known problem elements, but its surface approximates the surface mesh rather poorly (see Figure 3.5). To improve this approximation we iteratively optimize the volume mesh vertices as outlined in Algorithm 2. This minimizes the distance between the surface mesh and each vertex on the volume mesh's surface, and it maximizes the quality of all of the volume mesh's elements.

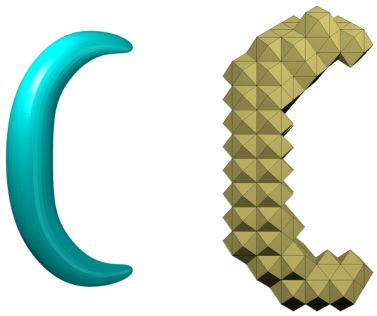

Figure 3.5. Given the crescent-shaped surface mesh on the left, our generated volume mesh is on the right as it stands after sculpting and before optimization.

Algorithm 2 Mesh optimization

compute volume mesh surface (V_{surf}, F), where $V_{\text{surf}} \subseteq V$
compute signed distance field d for S
while $\exists \mathbf{x}_i \in V$ s.t. $d(\mathbf{x}_i) > d_{\text{offset}}$ **do**
  for all $\mathbf{x}_i^n \in V$ **do**

$$\mathbf{x}_i^{n+1} \leftarrow \mathbf{x}_i^n + \left(\frac{1}{|\Omega_i|} \sum_{T_j \in \Omega_i} |T_j| \mathbf{c}_j \quad - \quad \mathbf{x}_i^n \right) \Delta t$$

  end for
  move all $\mathbf{x}_i^n \in V_{\text{surf}}$ toward $\mathbf{x}_i^{n+1}$ subject to continuous collision detection
end while

3.2.1 Vertex Updates

In every iteration we update the position of each vertex based on the tetrahedra incident to it. Our calculation for the new position is based on the work of Alliez and colleagues [3], who showed that the Optimal Delaunay Triangulation energy of Chen [11] for a given mesh, formulated as

$$E_{\text{ODT}} = \frac{1}{n+1} \sum_{i=1 \ldots N} \int_{\Omega_i} ||\mathbf{x} - \mathbf{x}_i||^2 d\mathbf{x} \tag{3.1}$$

is minimized by vertex positions

$$\mathbf{x}_i^* = \frac{1}{|\Omega_i|} \sum_{T_j \in \Omega_i} |T_j| \mathbf{c}_j \tag{3.2}$$

where Ω_i is the set of elements incident to vertex $\mathbf{x}_i$, $|\Omega_i|$ is the cumulative volume of the elements in Ω_i, and $\mathbf{c}_j$ is the circumcenter of T_j.

In contrast to Alliez and colleagues we do not move the vertices to these new positions. Instead we move them toward these positions by an amount proportional to a predetermined time step. This has the effect of iteratively drawing the surface of our volume mesh inwards while optimizing the quality of the elements in response.

This method of updating vertices produces results superior to those of our earlier experiments. At first we only updated the vertices on the surface of the volume mesh. We moved them in the direction of the negative gradient of the signed distance field described in Section 3.2.3. This resulted in poor-quality elements incident to those vertices. In response we tried a two-phase procedure, first maximizing proximity to the surface mesh and then optimizing the interior elements. This second phase involved moving vertices toward the average position of their adjacent vertices. While this did improve the mesh, it was not as effective as doing the simultaneous update described above.

3.2.2 Continuous Collision Detection

When we update the vertex positions in this way the surface of the volume mesh may eventually intersect the surface mesh. Because we desire an enclosing mesh we prevent this intersection by updating the vertices on the surface of the volume mesh separately from the other vertices. We treat these vertices and the tetrahedron edges between them as a surface that we advect, subject to continuous collision detection, toward the regular updated positions given by Equation (3.2).

Continuous collision detection is an a priori method that detects collisions between timesteps. It checks for two cases of collisions: one where a vertex passes through a triangle, and another where an edge passes through another edge. It is often used to simulate cloth and liquid interfaces, where a surface with many degrees of freedom can become irreparably tangled when using a posteriori collision detection methods. This makes it well-suited for our volume mesh surface, which also has many degrees of freedom. Our particular brand of continuous collision detection is that given by Brochu and Bridson [6], but we do not perform the improvements or topological changes that they describe.

3.2.3 Stopping Condition

We stop iterating when every vertex on the surface of the volume mesh is within some predefined distance d_{offset} of the surface mesh. We evaluate the distance between the surface mesh and a volume mesh vertex using a signed distance field representing the surface mesh. At the beginning of each iteration we evaluate the field at the position of each vertex on the surface of the volume mesh. If any of these values are greater than d_{offset} we carry out the current iteration.

We approximately represent this field by precomputing an octree of distance values that we construct based on the surface mesh, as do Bargteil and colleagues [4], allowing us to rapidly evaluate distances once per vertex in each iteration.

3.3 Implementation Details

In sculpting the initial volume mesh our second test for whether an element occupies any of the volume bounded by the surface mesh ostensibly runs once for each vertex in the surface mesh. Similarly the third test runs once for each triangle. We actually know beforehand that these tests will be negative for any vertex (or triangle) that lies outside the element's circumsphere. Therefore we only run the tests for vertices and triangles

that are not outside the element's circumsphere. This produces a significant speedup in our implementation, especially for surface meshes with many vertices and triangles.

In culling triangles an explicit triangle-sphere intersection test can be as expensive as the test that we are trying to obviate. Given our focus on coarse volume meshes for fine surface meshes, our triangles are so much smaller than our element circumspheres that we can simply check whether the triangle's centroid is inside the circumsphere.

CHAPTER 4

RESULTS AND DISCUSSION

We implemented the procedure described above and used our implementation program to generate volume meshes for four different surface meshes, shown in Figure 4.1. For each surface mesh we ran our program several times, each time starting with a different background lattice resolution and ending at a different offset distance. In the tables that follow we compare the output based on several measurements. Because a single, absolute offset distance is inappropriate for a set of volume meshes with different resolutions, we instead compare volume meshes with a common *ratio* between each one's offset distance d_{offset} and the width h of a grid cell in its initial lattice.

To portray mesh quality we report the minimum and maximum dihedral angles among all elements in each volume mesh, because the condition number of a finite-element computation on a given tetrahedral mesh depends on the lowest-quality tetrahedron in the volume mesh. To portray the degree to which the surface of the volume mesh approximates the surface mesh we also report the minimum, maximum, and mean distance from the surface mesh among all vertices on the surface of the volume mesh. We also report the ratio of the volume mesh's volume to the surface mesh's volume.

Our volume meshes for the banana surface, shown in Figure 4.2, mesh make clear the ability of our method to produce a set of viable volume meshes whose properties pre-

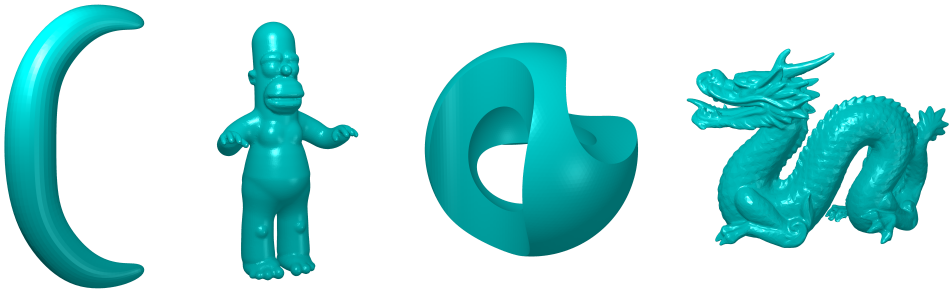

Figure 4.1. The four surface meshes we used as input for our tests. From left to right, they are called "banana," "Homer," "sculpture," and "dragon."

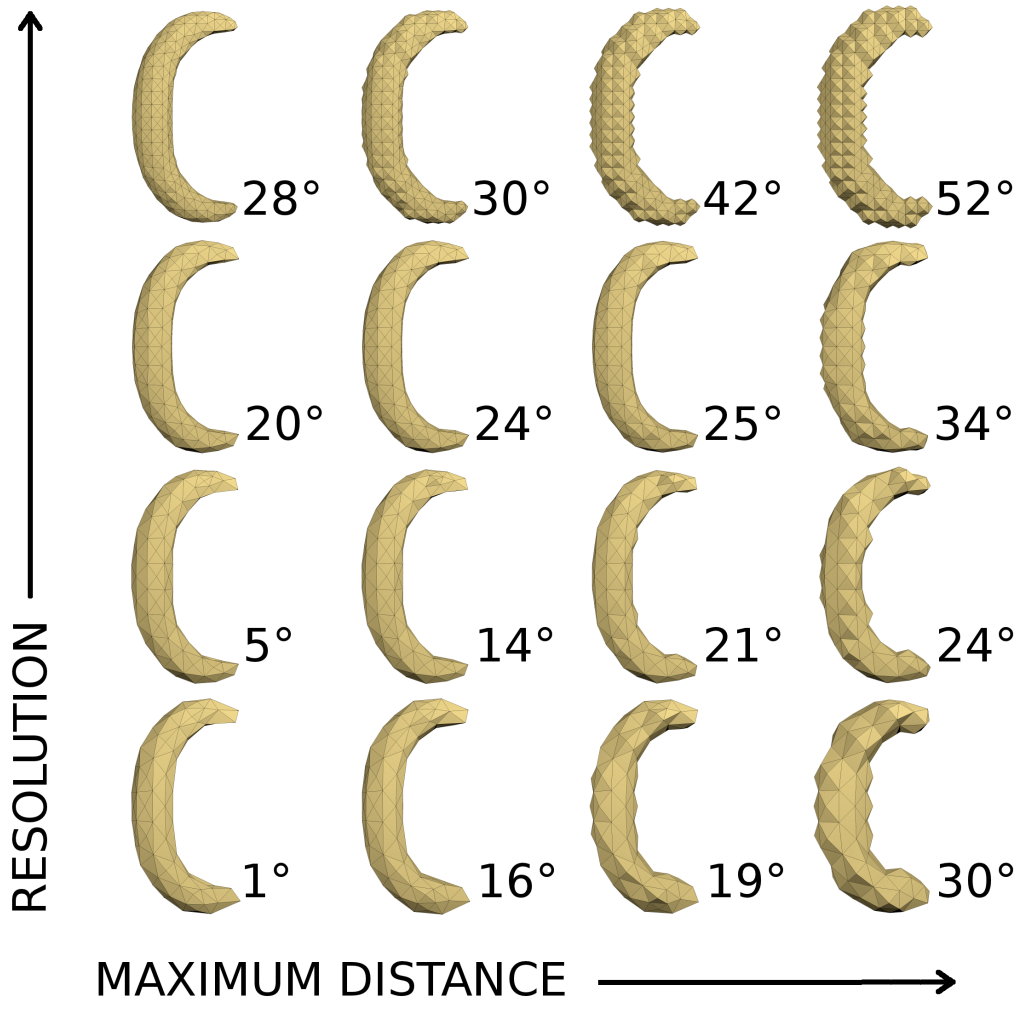

Figure 4.2. Our generated volume meshes for the banana surface mesh, each marked with the minimum dihedral angle of its worst element. The number of elements for each row is, from bottom to top, 447, 562, 912, and 2303. The offset band ratio for each column is, from left to right, 0.2, 0.4, 0.6, and 0.8.

dictably respond to changing parameters, providing a tradeoff between quality, efficiency, and proximity to the input. Except for one case, the meshes' minimum dihedral angles increase as the mesh resolution increases. In all cases their minimum angles increase as the offset distance is increased: this behavior is seen in all of our examples. The banana example also demonstrates that our method handles cases of concavity and negative curvature in the surface mesh.

The Homer surface is a more practical test case. It represents a human character, and volume meshes of human-shaped surface meshes are often in demand in computer animation. This surface has features of varying size, areas of varying curvature, and segments that are intuitively separate (those corresponding to the character's arms, legs, and head). Our method produces volume meshes that respect these aspects, as shown in Figure 4.3. They are recognizably human-shaped—notably preserving the arms and head as separate segments—and they approximate different parts of the surface equally well.

Homer brings out some idiosyncratic behavior in our method. While the generated volume meshes are of reasonably good quality and approximate the surface mesh reasonably well their quality measures do not vary with resolution in a straightforward way.

We can get a sense of why this happens by observing that the volume meshes with 556 elements generally capture more features of the surface mesh than do the volume meshes with 720 elements. We would expect the volume mesh with more elements to have more degrees of freedom, allowing its surface more opportunity to conform to the surface mesh. The meshes defy this expectation perhaps because the finer volume mesh has a greater number of elements that must approximate the same features, hindering each other's ability to do so. If this has such an unexpected effect on how well the volume mesh's surface approximates the surface mesh, then it is conceivable that it also has unexpected effects on element quality.

The sculpture surface mesh is interesting because it has sharp edges and nonzero genus. Our method handles these properties automatically, as shown in Figure 4.4. Our sculpting strategy described in Section 3.1 requires no special case for holes or interior hollows like those of the sculpture. The surface of our volume meshes can faithfully match the sharp edges of the surface mesh without introducing degenerate elements.

The dragon surface mesh has myriad small features and an intuitive shape (that of a snake) that is compressed into a different shape. Our method handles the former

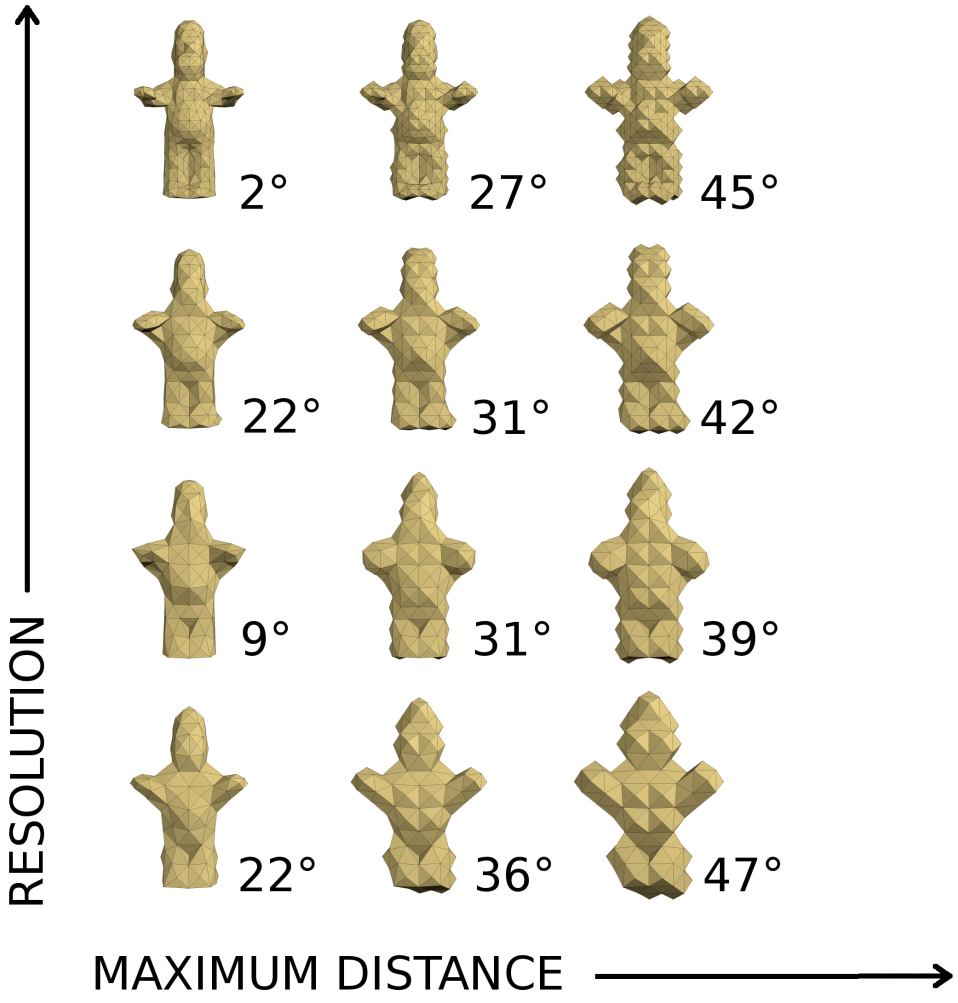

Figure 4.3. Our generated volume meshes for the Homer surface mesh, each marked with the minimum dihedral angle of its worst element. The number of elements for each row is, from bottom to top, 556, 720, 958, and 2497. The offset band ratio for each column is, from left to right, 0.4, 0.6, and 0.8.

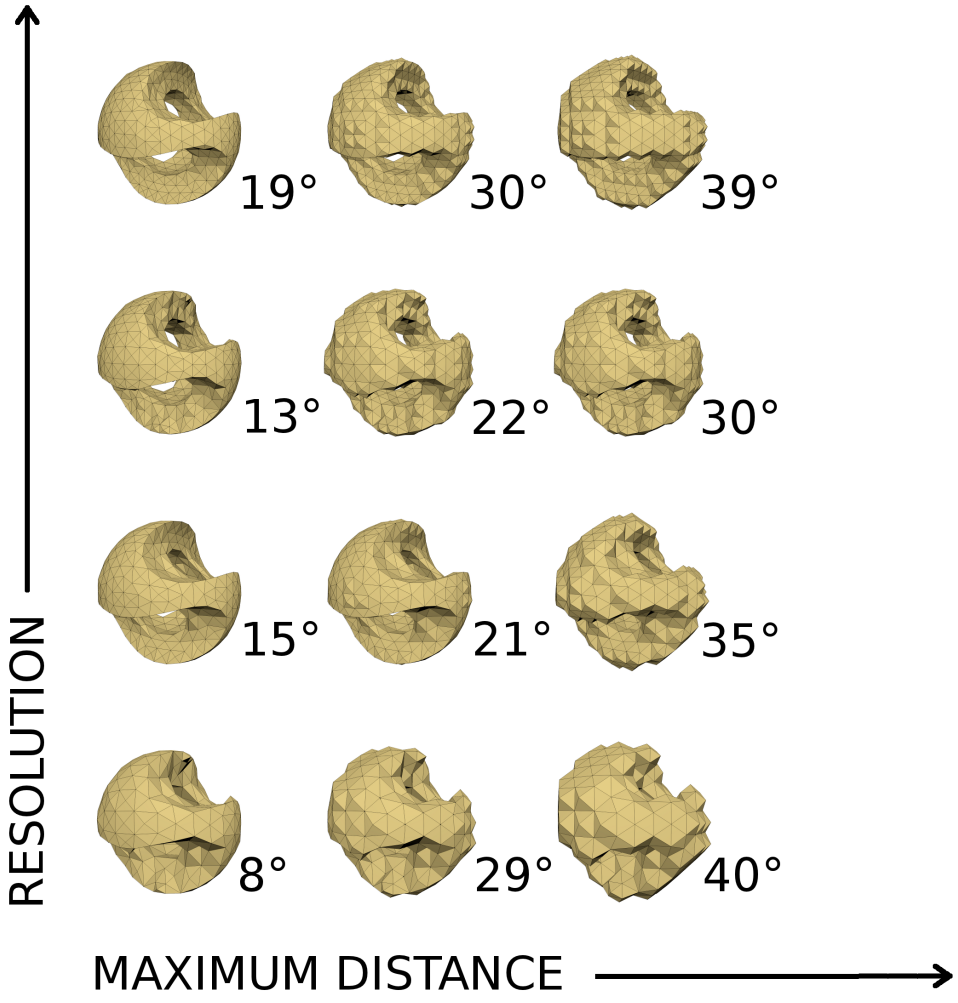

Figure 4.4. Our generated volume meshes for the sculpture surface mesh, each marked with the minimum dihedral angle of its worst element. The number of elements for each row is, from bottom to top, 1824, 3822, 4767, and 6204. The offset band ratio for each column is, from left to right, 0.4, 0.6, and 0.8.

aspect, generating volume meshes, shown in Figure 4.5, that conform to the surface mesh's concavities to a degree that varies continuously with the resolution of our initial lattice. It does not automatically address the latter aspect. The closely positioned but distantly connected parts of the surface mesh, such as the dragon's mouth, are smaller than our target resolution, and we do not capture them. For lesser offset distances this can lead to nonconvergence. Our collision detection prevents the vertices of large elements from reaching into small gaps, leaving them forever farther than d_{offset} away from the surface mesh in these areas. Given a reasonable offset distance, however, we can produce a volume mesh of good quality that still displays those features of the dragon that can be sampled at the volume mesh's resolution.

This vulnerability of our method—that it ignores thin gaps between intuitively separate segments—is a topic ripe for future investigation. One way to mitigate this, aside from increasing the resolution to inefficient levels, is to allow a user to mark certain segments as separate. The user could specify a plane section that intersects certain elements in the mesh before the optimization phase begins. We could then duplicate those elements and disconnect the duplicates, potentially allowing the optimization to separate them further. Other opportunities for future work include investigating the cause of the lower-quality elements in our worst volume meshes and devising a better prediction of volume mesh quality for a given surface mesh and resolution.

More detailed numerical measurements of our volume meshes for the banana, Homer, sculpture, and dragon surfaces appear in Tables 4.1, 4.2, 4.3, and 4.4, respectively.

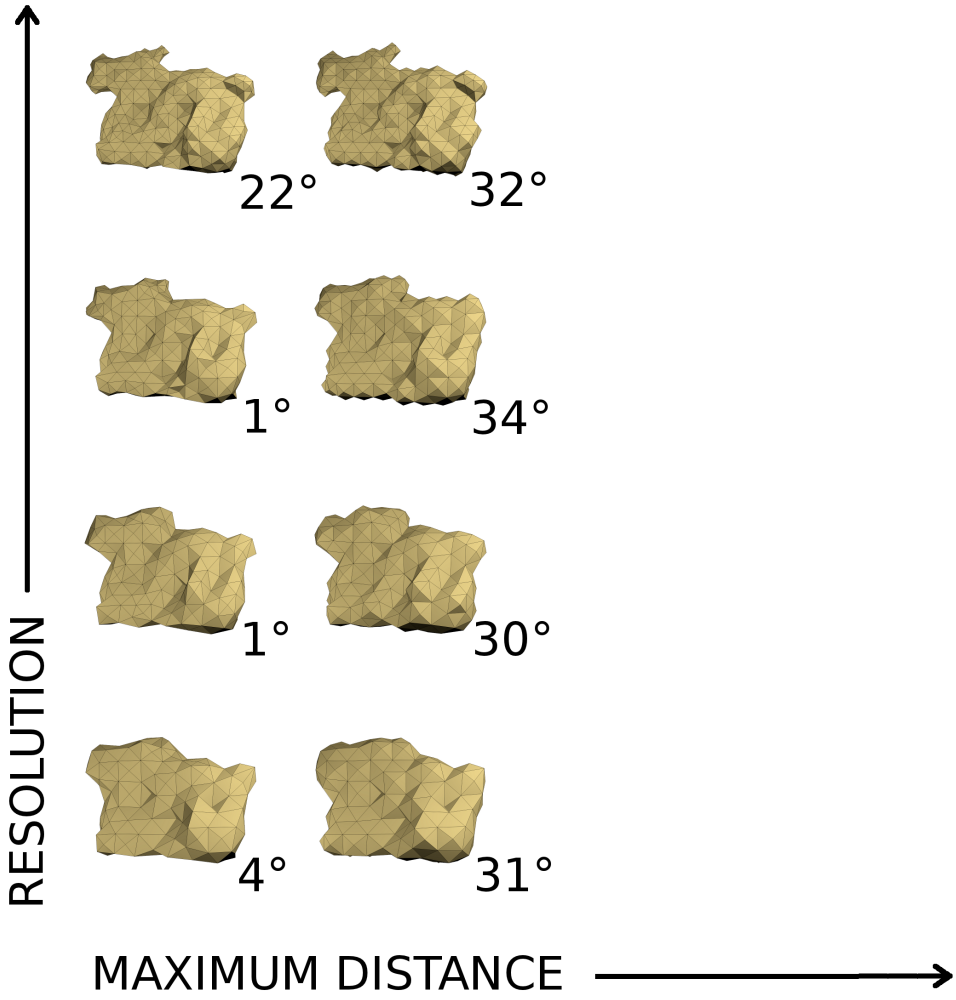

Figure 4.5. Our generated volume meshes for the dragon surface mesh, each marked with the minimum dihedral angle of its worst element. The number of elements for each row is, from bottom to top, 951, 1252, 1831, and 2834. The offset band ratio for each column is, from left to right, 0.6 and 0.8.

Table 4.1. Measurements of volume mesh results pictured in Figure 4.2. The offset band ratio for each column is, from left to right, 0.2, 0.4, 0.6, and 0.8.

banana, 2303 elements, $h = 0.5$				
d_{offset}	0.1	0.2	0.3	0.4
min. dihedral angle	28°	30°	42°	52°
max. dihedral angle	146°	141°	121°	102°
min. distance to surface	0.000457	0.00472	0.0000170	0.000100
max. distance to surface	0.0993	0.199	0.298	0.399
mean distance to surface	0.0201	0.0441	0.0806	0.126
volume ratio	1.06	1.16	1.33	1.59
banana, 912 elements, $h = 0.75$				
d_{offset}	0.15	0.3	0.45	0.6
min. dihedral angle	20°	24°	25°	34°
max. dihedral angle	155°	147°	145°	131°
min. distance to surface	0.00121	0.00100	0.00123	0.00186
max. distance to surface	0.149	0.299	0.449	0.598
mean distance to surface	0.0312	0.0393	0.0642	0.120
volume ratio	1.08	1.09	1.16	1.36
banana, 562 elements, $h = 1.0$				
d_{offset}	0.2	0.4	0.6	0.8
min. dihedral angle	5°	14°	21°	34°
max. dihedral angle	172°	159°	154°	137°
min. distance to surface	0.00103	0.00103	0.0148	0.0131
max. distance to surface	0.199	0.400	0.600	0.800
mean distance to surface	0.0447	0.0613	0.108	0.195
volume ratio	1.10	1.13	1.26	1.57
banana, 447 elements, $h = 1.25$				
d_{offset}	0.25	0.5	0.75	1.0
min. dihedral angle	1°	16°	19°	30°
max. dihedral angle	179°	143°	145°	122°
min. distance to surface	0.00102	0.0111	0.0146	0.0223
max. distance to surface	0.248	0.499	0.748	0.997
mean distance to surface	0.0557	0.0993	0.191	0.328
volume ratio	1.13	1.25	1.57	2.22

Table 4.2. Measurements of volume mesh results pictured in Figure 4.3. The offset band ratios for each column are, from left to right, 0.4, 0.6, and 0.8.

Homer, 2497 elements, $h = 0.2$			
d_{offset}	0.08	0.12	0.16
min. dihedral angle	2°	27°	45°
max. dihedral angle	176°	142°	112°
min. distance to surface	0.000100	0.000284	0.000151
max. distance to surface	0.0800	0.120	0.160
avg. distance to surface	0.00964	0.0230	0.0486
volume ratio	1.11	1.28	1.63
Homer, 958 elements, $h = 0.3$			
d_{offset}	0.12	0.18	0.24
min. dihedral angle	22°	31°	42°
max. dihedral angle	145°	138°	116°
min. distance to surface	0.000177	0.000179	0.00126
max. distance to surface	0.120	0.179	0.240
mean distance to surface	0.0283	0.0483	0.0748
volume ratio	1.32	1.52	1.84
Homer, 720 elements, $h = 0.35$			
d_{offset}	0.14	0.21	0.28
min. dihedral angle	9°	31°	39°
max. dihedral angle	162°	131°	112°
min. distance to surface	0.000136	0.00299	0.00076
max. distance to surface	0.140	0.209	0.279
mean distance to surface	0.0291	0.057	0.086
volume ratio	1.35	1.64	2.00
Homer, 556 elements, $h = 0.4$			
d_{offset}	0.16	0.24	0.32
min. dihedral angle	22°	36°	47°
max. dihedral angle	151°	135°	110°
min. distance to surface	0.00210	0.00867	0.0163
max. distance to surface	0.160	0.240	0.320
mean distance to surface	0.0397	0.0880	0.138
volume ratio	1.45	1.96	2.67

Table 4.3. Measurements of volume mesh results pictured in Figure 4.4. The offset band ratios for each column are, from left to right, 0.4, 0.6, and 0.8.

sculpture, 6204 elements, $h = 0.09$			
d_{offset}	0.036	0.054	0.072
min. dihedral angle	19°	30°	39°
max. dihedral angle	147°	132°	113°
min. distance to surface	0.00000163	0.00000740	0.000100
max. distance to surface	0.0359	0.0538	0.0720
avg. distance to surface	0.00372	0.00769	0.0135
volume ratio	1.11	1.20	1.33
sculpture, 4767 elements, $h = 0.10$			
d_{offset}	0.04	0.06	0.08
min. dihedral angle	13°	22°	30°
max. dihedral angle	152°	139°	129°
min. distance to surface	0.00000533	0.00000160	0.00000179
max. distance to surface	0.0398	0.0599	0.0799
mean distance to surface	0.00472	0.00812	0.0135
volume ratio	1.12	1.19	1.30
sculpture, 3822 elements, $h = 0.11$			
d_{offset}	0.044	0.066	0.088
min. dihedral angle	15°	21°	35°
max. dihedral angle	153°	144°	125°
min. distance to surface	0.00000429	0.00000431	0.000100
max. distance to surface	0.0440	0.0659	0.0879
mean distance to surface	0.00268	0.00621	0.0175
volume ratio	1.08	1.17	1.44
sculpture, 1824 elements, $h = 0.15$			
d_{offset}	0.06	0.09	0.12
min. dihedral angle	8°	29°	40°
max. dihedral angle	169°	131°	116°
min. distance to surface	0.000100	0.000100	0.000100
max. distance to surface	0.0599	0.0899	0.120
mean distance to surface	0.0117	0.0212	0.0328
volume ratio	1.33	1.50	1.76

Table 4.4. Measurements of volume mesh results pictured in Figure 4.5. The offset band ratios for each column are, from left to right, 0.6 and 0.8.

dragon, 2834 elements, $h = 1.25$		
d_{offset}	0.75	1.0
min. dihedral angle	22°	32°
max. dihedral angle	148°	130°
min. distance to surface	0.747	0.996
max. distance to surface	0.145	0.217
avg. distance to surface	0.378	0.556
volume ratio	1.46	1.63
dragon, 1831 elements, $h = 1.5$		
d_{offset}	0.9	1.2
min. dihedral angle	1°	34°
max. dihedral angle	180°	126°
min. distance to surface	0.899	1.20
max. distance to surface	0.120	0.275
mean distance to surface	0.00360	0.597
volume ratio	1.40	1.76
dragon, 1252 elements, $h = 1.75$		
d_{offset}	1.05	1.4
min. dihedral angle	1°	30°
max. dihedral angle	180°	133°
min. distance to surface	1.05	1.4
max. distance to surface	0.174	0.291
mean distance to surface	0.00166	0.519
volume ratio	1.51	1.76
dragon, 951 elements, $h = 2.0$		
d_{offset}	1.2	1.6
min. dihedral angle	4°	31°
max. dihedral angle	170°	136°
min. distance to surface	1.2	1.6
max. distance to surface	0.201	0.359
mean distance to surface	0.0713	0.534
volume ratio	1.60	1.95

CHAPTER 5

CONCLUSION

We have presented a procedure for tetrahedral mesh generation that is uniquely well suited to creating coarse, enclosing, high-quality volume meshes for animating arbitrary surface meshes in interactive simulations. Our procedure combines a pair of previously known techniques: using an initial background lattice and iteratively optimizing vertices. To these techniques we have contributed a new understanding of how BCC lattice-based meshes can respond to compression-like optimization, and we have demonstrated a way to improve that response. We have also contributed a method of simultaneously optimizing both the surface shape and the element quality of an enclosing volume mesh, obviating the need to define a detail-blurring offset surface. As our test results demonstrate, our procedure reliably generates volume meshes at levels of coarseness, shape approximation, and minimum element quality that are appropriate for interactive applications. It offers a tradeoff between these properties as well.

Our procedure is clearly an effective meshing strategy. Our implementation generated even the highest-resolution meshes in the span of a few minutes, so it is practical to use in real projects as well. Adoption of our procedure among simulation practitioners will depend on how often it fails, given their particular input surfaces, to converge to their required levels of shape approximation. Several avenues for improving its behavior in these cases already present themselves for future investigation. Therefore our procedure is a valuable addition to the literature of meshing for simulation, benefiting researchers and practitioners alike.

REFERENCES

[1] Acar, U. A., and Hudson, B. Dynamic mesh refinement with quad trees and off-centers. Tech. Rep. 121, Carnegie Mellon University School of Computer Science, 2007.

[2] Alim, U. R., Entezari, A., and Alim, T. M. The lattice-Boltzmann method on optimal sampling lattices. *IEEE Transactions on Visualization and Computer Graphics 15*, 4 (2009), 630–641.

[3] Alliez, P., Cohen-Steiner, D., Yvinec, M., and Desbrun, M. Variational tetrahedral meshing. In *Proceedings of ACM SIGGRAPH 2005* (2005).

[4] Bargteil, A. W., Goktekin, T. G., O'Brien, J. F., and Strain, J. A. A semi-Lagrangian contouring method for fluid simulation. *ACM Transactions on Graphics 25*, 1 (2006), 19–38.

[5] Bern, M., and Eppstein, D. Mesh generation and optimal triangulation. Tech. Rep. 47, Xerox Palo Alto Research Center, 1992.

[6] Brochu, T., and Bridson, R. Robust topological operations for dynamic explicit surfaces. *SIAM Journal on Scientific Computing 31*, 4 (2009), 2472–2493.

[7] Bronson, J., Levine, J., and Whitaker, R. Lattice cleaving: Conforming tetrahedral meshes of multimaterial domains with bounded quality. In *Proceedings of the 21st International Meshing Roundtable*. 2013, pp. 191–209.

[8] Buratynski, E. K. A fully automatic three-dimensional mesh generator for complex geometries. *International Journal for Numerical Methods in Engineering 30* (1990), 931–952.

[9] Capell, S., Green, S., Curless, B., Duchamp, T., and Popović, Z. Interactive skeleton-driven dynamic deformations. In *Proceedings of ACM SIGGRAPH 2002* (2002).

[10] Capell, S., Green, S., Curless, B., Duchamp, T., and Popović, Z. A multiresolution framework for dynamic deformations. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2002).

[11] Chen, L. Mesh smoothing schemes based on optimal Delaunay triangulations. In *Proceedings of the 13th International Meshing Roundtable* (2004), pp. 109–120.

[12] Cutler, B., Dorsey, J., and McMillan, L. Simplification and improvement of tetrahedral models for simulation. In *Proceedings of Eurographics Symposium on Geometry Processing 2004* (2004).

[13] Du, Q., and Wang, D. Tetrahedral mesh generation and optimization based on centroidal Voronoi tessellations. *International Journal for Numerical Methods in Engineering 56*, 9 (2003), 1355–1373.

[14] Edelsbrunner, H., Preparata, F. P., and West, D. B. Tetrahedrizing point sets in three dimensions. *Journal of Symbolic Computation 10*, 3-4 (1990), 335–347.

[15] Freitag, L. A., and Knupp, P. M. Tetrahedral mesh improvement via optimization of the element condition number. *International Journal for Numerical Methods in Engineering 53*, 6 (2002), 1377–1391.

[16] Freitag, L. A., and Ollivier-gooch, C. Tetrahedral mesh improvement using swapping and smoothing. *International Journal for Numerical Methods in Engineering 40*, 21 (1997), 3979–4002.

[17] Huang, J., Chen, L., Liu, X., and Bao, H. Efficient mesh deformation using tetrahedron control mesh. In *Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling* (2008), pp. 241–247.

[18] Hudson, B., Miller, G., and Phillips, T. Sparse Voronoi refinement. In *Proceedings of the 15th International Meshing Roundtable* (2006), pp. 339–356.

[19] Klingner, B. M., and Shewchuk, J. R. Aggressive tetrahedral mesh improvement. In *Proceedings of the 16th International Meshing Roundtable* (2007), pp. 3–23.

[20] Labelle, F., and Shewchuk, J. R. Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles. In *Proceedings of ACM SIGGRAPH 2007* (2007).

[21] Li, X.-Y., Teng, S.-H., and Üngör, A. Biting: Advancing front meets sphere packing. *International Journal for Numerical Methods in Engineering 49*, 1 (2000), 61–81.

[22] Liu, A., and Joe, B. Quality local refinement of tetrahedral meshes based on 8-subtetrahedron subdivision. *SIAM Journal on Scientific Computing 16*, 6 (1995), 1269–1291.

[23] Mitchell, S. A., and Vavasis, S. A. Quality mesh generation in higher dimensions. *SIAM Journal on Computing 29*, 4 (2000), 1334–1370.

[24] Molino, N., Bao, Z., and Fedkiw, R. A virtual node algorithm for changing mesh topology during simulation. In *Proceedings of ACM SIGGRAPH 2004* (2004).

[25] Molino, N., Bridson, R., Teran, J., and Fedkiw, R. A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In *Proceedings of the 12th International Meshing Roundtable* (2003), pp. 103–114.

[26] Mullen, P., Memari, P., de Goes, F., and Desbrun, M. HOT: Hodge-optimized triangulations. In *Proceedings of ACM SIGGRAPH 2011* (2011).

[27] Müller, M., and Gross, M. Interactive virtual materials. In *Proccedings of Graphics Interface 2004* (2004), pp. 239–246.

[28] Müller, M., Teschner, M., and Gross, M. Physically-based simulation of objects represented by surface meshes. In *Proceedings of Computer Graphics International 2004* (2004).

[29] Perucchio, R., Saxena, M., and Kela, A. Automatic mesh generation from solid models based on recursive spatial decompositions. *International Journal for Numerical Methods in Engineering 28*, 11 (1989), 2469–2501.

[30] Schöberl, J. NETGEN - An advancing front 2D/3D-mesh generator based on abstract rules. *Computing and Visualization in Science 1*, 1 (1997), 41–52.

[31] Shen, C., O'Brien, J. F., and Shewchuk, J. R. Interpolating and approximating implicit surfaces from polygon soup. In *Proceedings of ACM SIGGRAPH 2004* (2004).

[32] Shephard, M. S., and Georges, M. K. Automatic three-dimensional mesh generation by the finite octree technique. *International Journal for Numerical Methods in Engineering 32*, 4 (1991), 709–749.

[33] Shewchuk, J. R. Tetrahedral mesh generation by Delaunay refinement. In *Proceedings of the 14th Annual Symposium on Computational Geometry* (1998), pp. 86–95.

[34] Shewchuk, J. R. What is a good linear element? Interpolation, conditioning, and quality measures. In *Proceedings of the 11th International Meshing Roundtable* (2002), pp. 115–126.

[35] Si, H. TetGen, a quality tetrahedral mesh generator and three-dimensional Delaunay triangulator, v1.3 users manual. Tech. Rep. 9, Weierstrass Institute for Applied Analysis and Stochastics, 2004.

[36] Sifakis, E., Shinar, T., Irving, G., and Fedkiw, R. Hybrid simulation of deformable solids. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2007).

[37] Teran, J., Molino, N., Fedkiw, R., and Bridson, R. Adaptive physics based tetrahedral mesh generation using level sets. *Engineering with Computers 21*, 1 (2005), 2–18.

[38] Tournois, J., Srinivasan, R., and Alliez, P. Perturbing slivers in 3D Delaunay meshes. In *Proceedings of the 18th International Meshing Roundtable* (2009), pp. 157–173.

[39] Tournois, J., Wormser, C., Alliez, P., and Desbrun, M. Interleaving Delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. In *Proceedings of ACM SIGGRAPH 2009* (2009).

[40] Wojtan, C., and Turk, G. Fast viscoelastic behavior with thin features. In *Proceedings of ACM SIGGRAPH 2008* (2008).