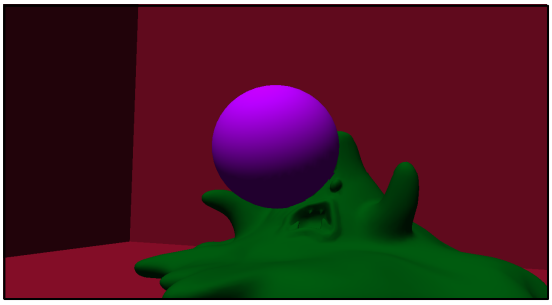
Automatic Construction of Coarse, High-Quality Tetrahedralizations that Enclose and Approximate Surfaces for Animation

David A. Stuart
University of Utah

Joshua A. Levine
Clemson University

Ben Jones
University of Utah

Adam W. Bargteil
University of Utah



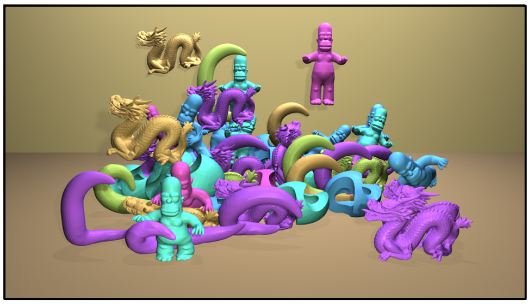


Figure 1: *Two simulations using meshes generated with our approach. Left: A real-time simulation of a projectile squashing a monster. Right: an offline simulation of a pile of objects.*

Abstract

Embedding high-resolution surface geometry in coarse control meshes is a standard approach to achieving high-quality computer animation at low computational expense. In this paper we present an effective, automatic method for generating such control meshes. The resulting high-quality, tetrahedral meshes enclose and approximate an input surface mesh, avoiding extrapolation artifacts and ensuring that the resulting coarse volumetric meshes are adequate collision proxies. Our approach comprises three steps: we begin with a tetrahedral mesh built from the body-centered cubic lattice that tessellates the bounding box of the input surface; we then perform a *sculpting* phase that carefully removes elements from the lattice; and finally a variational vertex adjustment phase iteratively adjusts vertex positions to more closely approximate the surface geometry. Our approach provides explicit trade-offs between mesh quality, resolution, and surface approximation. Our experiments demonstrate the technique can be used to build high-quality meshes appropriate for simulations within games.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.6.8 [Simulation and Modeling]: Types of Simulation—Animation; I.3.5 [Computer Graphics]: Computation Geometry and Geometric Modeling—Geometric Algorithms.

Keywords: mesh generation, animation of deformable bodies, finite element methods

1 Introduction

Finite element simulations of elastic bodies are commonplace in cinema and are being used increasingly in video games, e.g. Pixelux's Digital Molecular Matter (DMM) [Parker and O'Brien 2009]. One of the key differences in the requirements for game environments is the need for real-time finite element simulation. To achieve this computational efficiency, a coarse volumetric mesh is used to approximate an embedded, high-resolution surface mesh. The surface mesh is only queried for the purposes of rendering, and the coarse volumetric mesh is used for all other aspects of simulation. The ideal volumetric mesh should (1) enclose the surface, (2) closely approximate the surface, and (3) have high quality elements. Enclosing the surface is critical to avoiding extrapolation artifacts when computing the location of the surface. Close approximation of the input geometry is necessary if we are to use the coarse volumetric mesh for collision detection, which greatly simplifies and speeds the simulation compared with other embedding techniques (e.g. [Sifakis et al. 2007; Wojtan and Turk 2008]). Finally, high-quality elements ensure good numerical conditioning and robustness.

In this paper, we address the problem of automatically creating such volumetric meshes. Specifically, we seek coarse, high-quality volumetric meshes that both enclose an input surface and that have vertices lying no farther from the surface than a specified distance. Our solution to this problem begins with a tetrahedralization of a portion of the body-centered cubic (BCC) lattice and then iteratively moves the vertices toward a weighted average of the circumcenters of incident tetrahedra while constraining the volumetric mesh to enclose the surface. These steps move them closer to the surface while maintaining high quality in terms of the dihedral angles of the tetrahedral elements.

The key insight that enables our approach to be successful is that as vertices move toward the surface, any tetrahedron that has two faces that are on the surface of the volumetric mesh will flatten as it approaches the high resolution surface mesh, resulting in a sliver—a very poor-quality tetrahedron with four nearly co-planar points. We explicitly avoid creating such tetrahedra, preventing them during the *sculpting* phase where we cull tetrahedra from the background BCC lattice.

In our experiments this simple approach works extremely well. Compared with the alternative approach of computing an offset surface and running an off-the-shelf meshing program, we obtain superior quality, coarser meshes with much lower human effort. The

approach allows the user to tune the resolution through specification of the background lattice grid-spacing and provides a tradeoff between element quality and surface approximation through specification of the maximum surface-to-vertex distance. Our meshes are appropriate for fast finite element simulation of elastic bodies (see Figure 1).

2 Related Work

Tetrahedral tessellation of a given volume is a classic problem in computational geometry, inspiring an entire sub-community, with direct and important applications to computer animation. The general technique of embedding high-resolution geometry in low-resolution meshes dates back at least to 1986 when Sederberg and Parry introduced free-form deformations [Sederberg and Parry 1986]. More than a decade ago, Capell and colleagues [2002a] demonstrated the utility of embedded techniques for computer animation and argued [Capell et al. 2002b] that a volumetric mesh that closely fits the embedded surface mesh allows for more accurate deformations than a regular grid like that employed by Müller and colleagues [2004].

Since then the embedding technique has repeatedly appeared as a practical method of deforming and fracturing surface meshes [Molino et al. 2004; Sifakis et al. 2007; Wojtan and Turk 2008; Huang et al. 2008]. The efficacy of a coarse volumetric mesh for this technique also became apparent during this time [Capell et al. 2002a; Cutler et al. 2004; Müller and Gross 2004; Wojtan and Turk 2008]. Research on this technique has reaffirmed the perennial desirability of high-quality elements [Cutler et al. 2004; Molino et al. 2004]. Molino and colleagues [2004], Sifakis and colleagues [2007], and Wojtan and Turk [2008] use meshes that do not closely approximate the input geometry. To do so, they perform collisions with the high-resolution geometry and adjust the mass of partially filled elements. Nesme and colleagues [2009] went a step further and adjusted the finite element basis functions to account for partially filled elements. We advocate a simpler and faster approach—by ensuring that the coarse volumetric mesh is a close match to the underlying geometry, we can use the volumetric mesh for collision detection and do not need to adjust masses or basis functions. This approach can lead to significant computational savings in collision detection if the coarse mesh is much coarser than the high-resolution surface.

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" [2002]. Some are in common use among graphics researchers, like the ratio of the element's circumradius to its shortest edge [Shewchuk 1998], and some have less cachet, such as the condition number of the linear transformation between a unit equilateral tetrahedron and the element in question [Freitag and Knupp 2002]. 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 [Alliez et al. 2005]. Advancing front techniques are straightforward to implement, but they lack any guarantees of element quality [Li et al. 2000]. They can produce low-quality elements on the surface at sharp corners [Bern and Eppstein 1992] or in the interior where two advancing fronts meet. This approach is the basis of the popular "NETGEN" program [Schöberl 1997], 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 [Alliez et al. 2005]. This is the motivation for Delaunay-based algorithms like Shewchuk's Delaunay refinement [1998], Du and Wang's centroidal Voronoi tessellations [2003], Si's "TetGen" program [2004], and the CGAL 3D mesh generation package [cga]. In one of the earliest examples of Delaunay refinement, Edelsbrunner and colleagues [1990] begin with an assumed set of points to triangulate. In contrast, choosing the best set of points is a significant corollary problem, and inserting extra points can improve the refinement process [Hudson et al. 2006; Tournois et al. 2009b]. 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 [Tournois et al. 2009a]. 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 [Alliez et al. 2005].

A third approach 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 [2003], the level-set technique of Teran and colleagues [2005], "isosurface stuffing" by Labelle and Shewchuk [2007], and "lattice cleaving" by Bronson and colleagues [2013]. 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 [Alliez et al. 2005]. 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 [Acar and Hudson 2007]. The octree approach is the foundation of the "QMG" technique given by Mitchell and Vavasis [2000], and it appears in older methods like those of Buratynski [1990], Perucchio and colleagues [1989], and Shephard and colleagues [1991].

Finally, an important fifth approach to mesh generation is physics-based vertex optimization. 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 [Du and Wang 2003], Optimal Delaunay Triangulations [Alliez et al. 2005], and Hodge-Optimized Triangulations [Mullen et al. 2011]. Both Molino and colleagues [2003] and Teran and colleagues [2005] 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 [1997] published a widely cited discussion 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 [1995], as well as more dramatic connectivity changes, as in the simplification method by Cutler and colleagues [2004] and Klingner and Shewchuk's improvement program "Stellar" [2007].

Procedures for generating volumetric 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 volumetric mesh is everywhere exterior to the surface mesh. As more general meshing algorithms usually do not guarantee this property, an enclosing volumetric 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 [2004] suggest this procedure as an application of their implicit surface generation algorithm. This procedure ensures an erroneous approximation of the input, since the offset surface loses the detail of the original surface.

3 Method

Our method takes as input a high-resolution surface mesh and generates a coarse, high-quality tetrahedral mesh that approximates this surface. For clarity we denote the input surface mesh as $\mathcal{S}$, the volumetric mesh as $\mathcal{V}$, and the boundary/surface of the volumetric mesh as $\partial\mathcal{V}$.

We generate the volumetric mesh in three phases (see Figure 2). First, we construct a tetrahedralization of the bounding box of the input surface. Second, in the *sculpting* phase we carefully remove tetrahedra that do not overlap the volume enclosed by the surface, ensuring that in the resulting mesh there are no tetrahedra with two faces on the surface. Third, we iteratively update vertex locations to better approximate the input geometry while maintaining high-quality tetrahedra.

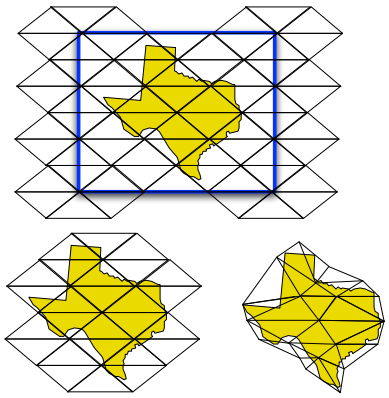


Figure 2: *A 2D illustration of our approach, which comprises three phases: Initial tetrahedralization of the bounding box (shown in blue), sculpting, and variational vertex adjustment. Note that the initial mesh is a union of* snowflakes *(see Section 3.2)*

3.1 Tetrahedralizing the Bounding Box

We generate an initial tetrahedral volumetric 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).

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° [Labelle and Shewchuk 2007]. Its vertices are an optimal sampling of the volume for representing trivariate functions like deformation forces [Alim et al. 2009]. 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 topology-changing operations.

The user chooses the resolution of the BCC lattice, allowing (indirect) control over the resolution of the final mesh. Because we do not allow topological operations, any topology of the high-resolution surface that is not resolved by the BCC lattice will be lost.

3.2 Sculpting

At this point the volumetric 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 iterate over the elements, determining whether each element occupies some of the bounded volume using three geometric tests illustrated in Figure 4. If an element intersects the bounded volume we do not mark it for removal.

The first test checks whether any vertex of the element is inside the bounded volume. If so, we clearly do not want to remove the element from the mesh. To speed this computation we build a distance field for the input surface (using the method of Bargteil and colleagues [2006]).

However, not all elements that intersect the bounded volume will have vertices inside the surface, as illustrated in Figure 4. Our second test, performed for each tetrahedron that has all of its vertices outside the surface, checks if there are any surface vertices inside the tetrahedron. This requires computing four dot products for every surface mesh vertex. This approach is not particularly efficient, having cost proportional to $|V||T|$, with $|V|$ surface vertices and $|T|$ tetrahedra. We do perform a simple culling mechanism, first checking that surface vertices lie inside the tetrahedra's circumsphere and more advanced culling is certainly possible. However, this test accounts for only a small fraction of our computation time.

It is still possible that the tetrahedron intersects the bounded volume, especially if the input triangle mesh is relatively coarse. Our final test checks every edge of the element for intersection with every triangle. Again, we cull triangles outside the element's circumsphere.

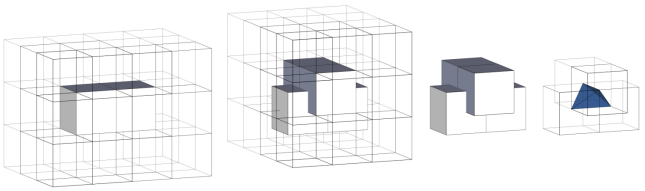


Figure 3: *The tetrahedra of the BCC lattice. The lattice is constructed from two grids, and the points of each grid lie at the cell centers of the other grid. The tetrahedra are all like the one shown at right: in both grids, two of the tetrahedron's vertices are at the centers of adjacent cells.*

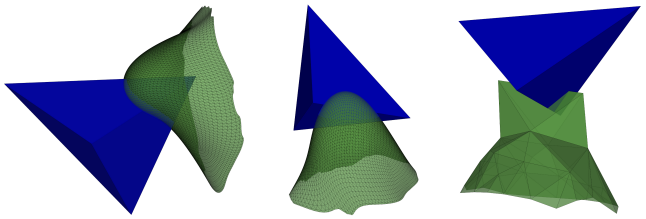


Figure 4: *Three cases of an element occupying part of the volume bounded by the surface mesh. On the left, a vertex of the element is inside the bounded volume. In the center, a vertex of the surface mesh is inside the element. On the right, an edge of the element intersects two triangles of the surface mesh.*

Defusing the Bombs In our initial experiments using the sculpting procedure outlined above and our variational vertex adjustment procedure described in Section 3.3, we noted that the poorly shaped elements had one thing in common: a pair of adjacent faces on the surface of the tetrahedral mesh ($\partial\mathcal{V}$). That is, two adjacent faces on the surface were incident to only one tetrahedron. An example is shown in Figure 5. As we adjust these elements' vertices and move them toward the input surface ($\mathcal{S}$) they *flatten*, becoming *slivers*—low-quality elements with four nearly co-planar vertices. We would like our mesh to contain no such *bomb* elements that predictably turn into slivers later. We accomplish this by ensuring that the result of the sculpting phase is a union of *snowflakes*—sets of 24 elements incident to a common vertex. One is illustrated in Figure 5. It is easy to show that a snowflake contains no bomb elements and that any union of snowflakes preserves this property.

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 adding back elements such that all bombs are part of complete snowflakes. Specifically, for each bomb element we add back all removed elements that are incident to one of its vertices. This step ensures that the snowflake centered at that vertex—one of the snowflakes that contains the bomb element—is part of the mesh. We could choose one vertex of the bomb arbitrarily and add back all incident elements, but this greedy approach may not be optimal. Instead we perform a global search that considers all possible combinations of snowflakes, one per bomb, and choose the one that

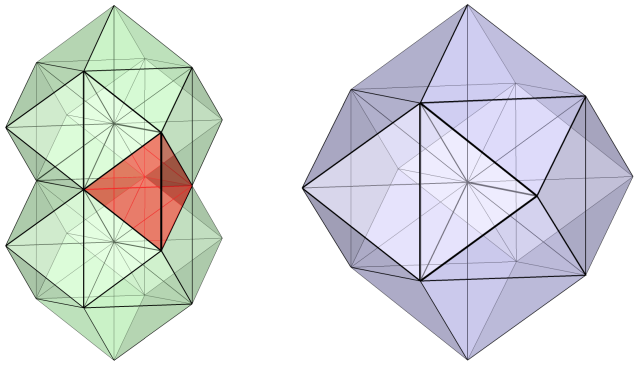


Figure 5: *Left: An initial mesh with a single bomb element. The bomb element, in front and darkly colored, has two faces on the surface of the mesh. The other elements, lightly colored, each have at most one face on the surface. Right: A snowflake. Each element in this set is incident to the vertex in the center.*

results in the least number of elements. As there are four choices for each bomb element, if there are n bomb elements, there are n^4 possibilities. Given that there are generally few bomb elements, this approach is quite tractable.

We also considered two alternative strategies to avoiding bomb elements. Before sculpting, the mesh is a union of snowflakes centered on vertices that all lie in one of the two staggered grids of the BCC lattice. Thus, one can restrict the choice to these snowflakes. This approach does indeed lead to a mesh that is a union of snowflakes and contains no bomb elements, but it contains far more elements than our approach and results in a final mesh that is lower quality. We also attempted to remove bomb elements by subdividing each bomb element into four elements, each incident to the bomb element's centroid. This also 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 volumetric mesh. The faces of the original bomb element were still pressed into a sliver even though the sliver was subdivided.

3.3 Variational Vertex Adjustment

At this point we have an enclosing volumetric mesh free of known problem elements, but the surface of this volumetric mesh ($\partial\mathcal{V}$) approximates the high-resolution surface mesh ($\mathcal{S}$) rather poorly. To improve this approximation we iteratively adjust the positions of the volumetric mesh vertices. This decreases the distance between the surface mesh and each vertex on the volumetric mesh's surface while maintaining high-quality tetrahedral elements. Note that we adjust the positions of *all* mesh vertices, allowing interior vertices to adjust to updates of surface vertices.

Vertex Updates In every iteration we update the position of each vertex based on the incident tetrahedra. Our calculation for the new position is based on the work of Alliez and colleagues [2005], who showed that the Optimal Delaunay Triangulation energy of Chen [2004] for a given mesh, formulated as

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

is minimized by vertex positions

$$\mathbf{x}_i^* = \frac{1}{|\Omega_i|} \sum_{T_j \in \Omega_i} |T_j| \mathbf{c}_j \tag{2}$$

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. Essentially, this is a volume-weighted average of the circumcenters of incident tetrahedra.

In contrast to Alliez and colleagues we do not move the vertices directly to these 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 volumetric mesh inwards while maintaining the quality of the elements in response.

This vertex update produces superior results to alternatives we considered: the simpler approach of moving vertices along the negative gradient of the distance field and a more complicated, two-phase process that first moved vertices along the negative gradient to the distance field and then applied smoothing to improve quality.

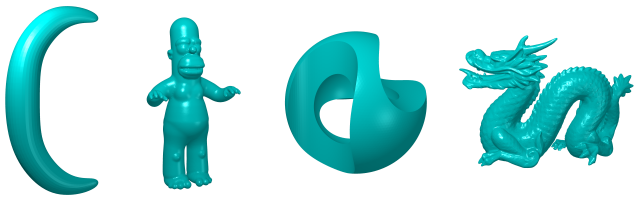


Figure 6: *The four surface meshes we used as input for our tests. From left to right, they are called "banana," "humanoid," "sculpture," and "dragon."*

Continuous Collision Detection With our vertex update rule, the surface of the volumetric mesh ($\partial\mathcal{V}$) may intersect the surface mesh ($\mathcal{S}$). Because we desire a volumetric mesh ($\mathcal{V}$) that encloses the input surface ($\mathcal{S}$), we prevent such intersections by treating the vertices on the surface of the volumetric mesh ($\partial\mathcal{V}$) specially. Specifically, we subject the triangle mesh that represents the surface of the volumetric mesh ($\partial\mathcal{V}$) to continuous collision detection against the (unmoving) input surface mesh ($\mathcal{S}$). We use the *El Topo* library [Brochu and Bridson 2009], setting the mass of vertices in the input surface mesh ($\mathcal{S}$) to the highest value available on our machine.

Stopping Condition We continue updating vertex locations until every vertex on the surface of the volumetric mesh ($\partial\mathcal{V}$) is within some predefined distance d_{offset} of the surface mesh ($\mathcal{S}$). At the beginning of each iteration we evaluate the distance of every vertex in $\partial\mathcal{V}$ to the surface, $\mathcal{S}$, with a query to our distance field. If any of these values are greater than d_{offset} we carry out the current iteration.

We additionally experimented with alternative stopping conditions and various multi-phase vertex adjustments, such as moving vertices more slowly as they approached the surface. None of these approaches produced improved results.

3.4 Perturbation

The results of our approach are extremely sensitive to perturbations of the background lattice, as perturbations will lead to different tetrahedra being removed during sculpting. We can use this fact to our advantage—by perturbing the background lattice with random translations and rotations we can generate several candidate meshes and select the best one. This approach can have a dramatic affect on final quality, albeit at additional cost. As mesh generation is generally a pre-process this additional cost is of little consequence. Moreover, this sampling process can be stopped at any time, returning the best mesh found so far.

4 Results and Discussion

We have implemented our approach described above and tested it on four different surface meshes, shown in Figure 6. 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. For each combination of these parameters we ran the software multiple times (between 3 and 13 times depending on the example) with offset and rotated background grids and chose the best resulting output. Because each run takes but a few minutes even with our unoptimized implementation, multiple runs are quite practical and can provide significantly improved results over single runs. When our method converges, individual runs were all less than ten minutes for a single thread on modern hardware.

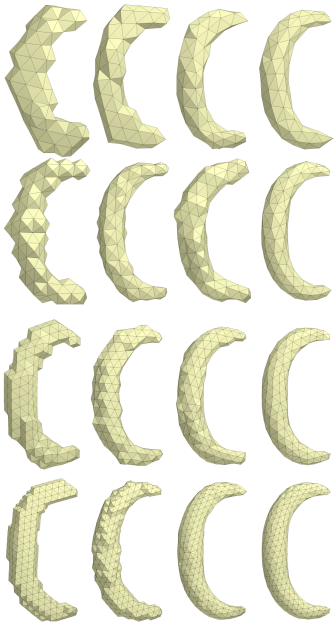


Figure 7: *Our generated volumetric meshes for the banana surface mesh. From top to bottom the meshes are sculpted from a lattice of increasing resolution, and from left to right they are constrained by a decreasing offset band ratio.*

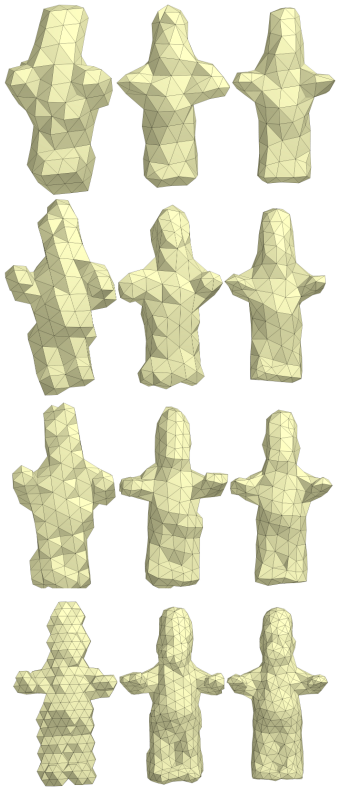


Figure 8: *Our generated volumetric meshes for the humanoid surface mesh. From top to bottom the meshes are sculpted from a lattice of increasing resolution, and from left to right they are constrained by a decreasing offset band ratio.*

In Table 1 we compare the output based on several measurements. A single, absolute offset distance is inappropriate for a set of volumetric meshes with different resolutions, so we instead compare volumetric 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 volumetric mesh, because the condition number of a finite-element computation on a given tetrahedral mesh depends on the lowest-quality tetrahedron in the volumetric mesh. To portray the degree to which the surface of the volumetric mesh ($\partial\mathcal{V}$) approximates the surface mesh ($\mathcal{S}$) we also report the minimum, maximum, and mean distance from the surface mesh ($\mathcal{S}$) among all vertices on the surface of the volumetric mesh ($\partial\mathcal{V}$). We also report the ratio of the volumetric mesh's volume to the surface mesh's volume.

Our volumetric meshes for the banana surface mesh, shown in Figure 7, make clear the ability of our method to produce a set of viable volumetric meshes whose properties predictably respond to changing parameters, providing a tradeoff between quality, efficiency, and proximity to the input. Except for three cases, 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 humanoid surface is a more practical test case. 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 volumetric meshes that respect these aspects, as shown in Figure 8. They are recognizably human-shaped—notably preserving the arms and head as separate segments—and they approximate different parts of the surface equally well. However, our approach did not separate the legs.

The sculpture surface mesh is interesting because it has sharp edges and nonzero genus. Our method handles these properties automatically, as shown in Figure 9. Our sculpting strategy described in Section 3 requires no special case for holes or interior hollows like those of the sculpture, provided that the resolution is high enough that they are resolved on the background grid. The surface of our volumetric 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 aspect, generating volumetric meshes, shown in Figure 10, 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 volumetric mesh of good quality that still displays those features of the dragon that can be sampled at the volumetric mesh's resolution.

We also compare our approach to the alternative: computing an offset surface and using off-the-shelf meshing software. Our underlying hypothesis is that by giving the mesher the freedom to choose the offset surface we can achieve higher quality meshes. Our experiments, detailed in Table 2, bear this out. Our mesher, Cetra, outperforms the competition, and at much lower human effort.

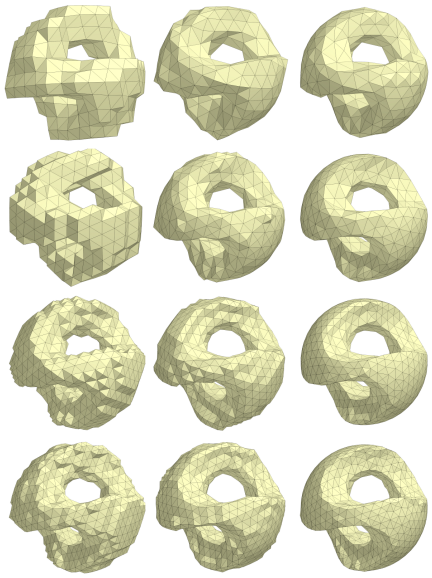


Figure 9: *Our generated volumetric meshes for the sculpture surface mesh. From top to bottom the meshes are sculpted from a lattice of increasing resolution, and from left to right they are constrained by a decreasing offset band ratio.*

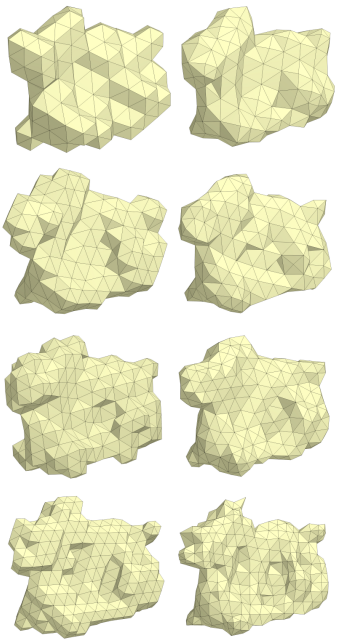


Figure 10: *Our generated volumetric meshes for the dragon surface mesh. From top to bottom the meshes are sculpted from a lattice of increasing resolution, and from left to right they are constrained by a decreasing offset band ratio.*

	banana, $h = 0.5$				banana, $h = 0.75$			
elements	2364	2320	2349	2291	890	891	950	911
min. angle	21°	26°	34°	45°	20°	25°	32°	44°
max. angle	144°	143°	128°	109°	135°	147°	123°	109°
min. dist.	0.000550	0.00197	4.11×10^{-10}	0.00000161	0.00238	0.000601	0.00673	0.000457
max. dist.	0.0993	0.199	0.291	0.391	0.150	0.277	0.432	0.583
mean dist.	0.0153	0.0279	0.0685	0.093	0.0326	0.0635	0.117	0.191
vol. ratio	1.04	1.09	1.28	1.40	1.07	1.17	1.39	1.74
	banana, $h = 1.0$				banana, $h = 1.25$			
elements	537	511	494	538	300	317	312	330
min. angle	20°	27°	33°	43°	21°	27°	37°	46°
max. angle	146°	139°	125°	116°	138°	135°	123°	113°
min. dist.	0.00253	0.0000415	0.021	0.0217	0.00244	0.0106	0.0352	0.0168
max. dist.	0.199	0.390	0.584	0.772	0.248	0.488	0.739	0.964
mean dist.	0.0472	0.121	0.192	0.299	0.0858	0.157	0.280	0.404
vol. ratio	1.10	1.33	1.61	2.14	1.18	1.40	1.94	2.61
	humanoid, $h = 0.2$				humanoid, $h = 0.3$			
elements	–	2468	2440	2406	–	927	979	945
min. angle	–	16°	28°	44°	–	20°	29°	45°
max. angle	–	160°	137°	112°	–	147°	131°	112°
min. dist.	–	0.000405	0.000281	0.0001	–	0.00277	0.000645	0.00206
max. dist.	–	0.0798	0.114	0.154448	–	0.118	0.174	0.240
mean dist.	–	0.0118	0.0208	0.0420	–	0.0231	0.0431	0.0790
vol. ratio	–	1.16	1.25	1.54	–	1.26	1.46	1.95
	humanoid, $h = 0.35$				humanoid, $h = 0.4$			
elements	–	722	672	694	–	504	556	526
min. angle	–	21°	35°	44°	–	21°	33°	42°
max. angle	–	141°	126°	110°	–	132°	126°	112°
min. dist.	–	0.000386	0.00625	0.00618	–	0.000706	0.00406	0.00520
max. dist.	–	0.146	0.209	0.260	–	0.159	0.236	0.294
mean dist.	–	0.0278	0.0633	0.0984	–	0.0431	0.0793	0.103
vol. ratio	–	1.32	1.71	2.16	–	1.48	1.93	2.26
	sculpture, $h = 0.13$				sculpture, $h = 0.15$			
elements	–	7972	7776	7860	–	5587	5467	5450
min. angle	–	12°	23°	32°	–	10°	26°	35°
max. angle	–	156°	145°	132°	–	163°	142°	123°
min. dist.	–	0.00000338	0.0000269	0.0000884	–	0.000000755	0.0000377	0.0000119
max. dist.	–	0.0459	0.0729	0.100	–	0.0548	0.0862	0.117
mean dist.	–	0.00471	0.00929	0.0158	–	0.00537	0.0139	0.0223
vol. ratio	–	1.08	1.16	1.26	–	1.09	1.22	1.35
	sculpture, $h = 0.11$				sculpture, $h = 0.25$			
elements	–	2730	2753	2714	–	1666	1666	1615
min. angle	–	10°	21°	39°	–	14°	22°	38°
max. angle	–	161°	152°	117°	–	151°	143°	123°
min. dist.	–	0.0000196	0.0001	0.0001	–	0.000237	0.000100	0.0000999
max. dist.	–	0.0756	0.116	0.158	–	0.100	0.144	0.200
mean dist.	–	0.00853	0.0195	0.0398	–	0.0180	0.0272	0.0520
vol. ratio	–	1.15	1.31	1.61	–	1.28	1.41	1.75
	dragon, $h = 1.25$				dragon, $h = 1.5$			
elements	–	–	2417	2488	–	–	1618	1603
min. angle	–	–	23°	39°	–	–	24°	40°
max. angle	–	–	141°	114°	–	–	143°	114°
min. dist.	–	–	0.000195	0.000728	–	–	0.0186	0.0192
max. dist.	–	–	0.740	0.967	–	–	0.865	1.17
mean dist.	–	–	0.157	0.282	–	–	0.211	0.357
vol. ratio	–	–	1.56	1.98	–	–	1.74	2.16
	dragon, $h = 1.75$				dragon, $h = 2.0$			
elements	–	–	1127	1141	–	–	887	847
min. angle	–	–	26°	37°	–	–	29°	42°
max. angle	–	–	140°	116°	–	–	138°	112°
min. dist.	–	–	0.00429	0.0457	–	–	0.000137	0.00252
max. dist.	–	–	1.04	1.34	–	–	1.19	1.55
mean dist.	–	–	0.259	0.405	–	–	0.334	0.526
vol. ratio	–	–	1.85	2.21	–	–	2.07	2.59

Table 1: *Measurements of volumetric mesh results. For each input and each lattice cell-width h, we created meshes using four different offset band ratios: from left to right, they are 0.2, 0.4, 0.6, and 0.8. Dashes represent cases in which our adjustment process did not converge.*

	NETGEN	TetGen	I. Stuffing	CGAL	Cetra
banana					
tets	1945	22154	8341	3625	478
min. °	1.90°	2.68°	13.6°	3.57°	23.6°
max. °	170°	170°	161°	174°	151°
humanoid					
tets	1377	15044	22906	7663	536
min. °	8.41°	2.78°	16.9°	0.956°	18.1°
max. °	167°	170°	156°	178°	151°
sculpture					
tets	–	26915	14880	7363	2785
min. °	–	0.526°	16.5°	1.03°	19.8°
max. °	–	175°	148°	178°	147°
dragon					
tets	1297	9648	45956	4696	1110
min. °	8.69°	3.96°	17.4°	2.648°	22.3°
max. °	166°	170°	155°	176°	135°

Table 2: *Comparisons between our method ("Cetra") and popular existing systems. NETGEN was unable to produce a mesh for the sculpture surface.*

To demonstrate the effectiveness of our method in practice, we produced animations of surfaces embedded in volumetric meshes that we created with our mesher. We deformed the meshes using an elastic deformation simulator based on the finite-element method. The first animation is a recording of a computer game running this simulator in real time. It depicts, in first-person perspective, a player throwing projectiles at a monstrous pile of slime, which deforms based on the trajectory that the player chooses. We see here that by using our mesher we can simulate deformation quickly enough to respond to unpredictable user input.

In contrast, we created the second animation under predetermined conditions and took several time steps in the simulator to produce each frame. This animation shows many deformable objects falling into an invisible box, pressing against each other and the box's sides. The objects' simulated contacts occur simultaneously with their apparent contacts: there is no intersection, and neither are there unrealistic gaps between colliding objects. Our meshes are of high enough quality that the simulation stays stable. One object in particular, despite undergoing extreme deformation, slides across the frame and returns to normal without causing problems. In addition to these animations of physics-based simulation, we animated our process of variational vertex adjustment. Here we see the tetrahedra change in quality as they better approximate the input surfaces, taking the form of the inputs while maintaining high minimum dihedral angles.

Limitations and Future Work A key limitation of our method—that it ignores small topological features such as thin gaps between intuitively separate segments—is a topic ripe for future investigation. One way to address this limitation, aside from increasing the resolution to inefficient levels, is to allow a user to mark certain segments as separate. We could then duplicate elements around these segments and disconnect the duplicates, potentially allowing the adjustments to separate them further. Other opportunities for future work include investigating the cause of the lower-quality elements in our worst volumetric meshes and devising a better prediction of volumetric mesh quality for a given surface mesh and resolution.

Conclusion We have presented a method for tetrahedral mesh generation that is uniquely well suited to creating coarse, enclosing, high-quality volumetric meshes for animating arbitrary surface meshes. Our method combines two previously known techniques: using an initial BCC lattice and variational vertex adjustments. However, the success of our approach hinges on a critical insight—that tetrahedra in the initial lattice that have two adjacent faces on the surface become slivers under vertex adjustments in the presense of collision detection. We solve this issue by ensuring that our initial mesh is a union of *snowflakes*. Moreover, our approach compares favorably with the tedious approach of creating an offset surface and using an off-the-shelf meshing algorithm.

Acknowledgements

The authors wish to thank the anonymous reviewers for their helpful comments. We also thank Dan Gerszewski and Jonathan Shewchuk for their help with our comparison experiments. This work was supported in part by a gift from Adobe Systems Incorporated and National Science Foundation Awards CNS-0855167, IIS-1249756, and IIS-1314896.

References

ACAR, U. A., AND HUDSON, B. 2007. Dynamic mesh refinement with quad trees and off-centers. Tech. Rep. 121, Carnegie Mellon University School of Computer Science.

ALIM, U. R., ENTEZARI, A., AND MÖLLER, T. 2009. The lattice-Boltzmann method on optimal sampling lattices. *IEEE Transactions on Visualization and Computer Graphics 15*, 4, 630–641.

ALLIEZ, P., COHEN-STEINER, D., YVINEC, M., AND DESBRUN, M. 2005. Variational tetrahedral meshing. *ACM Trans. Graph. 24*, 3 (July), 617–625.

BARGTEIL, A. W., GOKTEKIN, T. G., O'BRIEN, J. F., AND STRAIN, J. A. 2006. A semi-Lagrangian contouring method for fluid simulation. *ACM Transactions on Graphics 25*, 1, 19–38.

BERN, M., AND EPPSTEIN, D. 1992. Mesh generation and optimal triangulation. Tech. Rep. 47, Xerox Palo Alto Research Center.

BROCHU, T., AND BRIDSON, R. 2009. Robust topological operations for dynamic explicit surfaces. *SIAM Journal on Scientific Computing 31*, 4, 2472–2493.

BRONSON, J., LEVINE, J., AND WHITAKER, R. 2013. Lattice cleaving: Conforming tetrahedral meshes of multimaterial domains with bounded quality. In *Proceedings of the 21st International Meshing Roundtable*. 191–209.

BURATYNSKI, E. K. 1990. A fully automatic three-dimensional mesh generator for complex geometries. *International Journal for Numerical Methods in Engineering 30*, 931–952.

CAPELL, S., GREEN, S., CURLESS, B., DUCHAMP, T., AND POPOVIĆ, Z. 2002. Interactive skeleton-driven dynamic deformations. *ACM Trans. Graph. 21*, 3 (July), 586–593.

CAPELL, S., GREEN, S., CURLESS, B., DUCHAMP, T., AND POPOVIĆ, Z. 2002. A multiresolution framework for dynamic deformations. In *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 41–47.

CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

CHEN, L. 2004. Mesh smoothing schemes based on optimal Delaunay triangulations. In *Proceedings of the 13th International Meshing Roundtable*, 109–120.

CUTLER, B., DORSEY, J., AND MCMILLAN, L. 2004. Simplification and improvement of tetrahedral models for simulation. In *Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing*, 93–102.

DU, Q., AND WANG, D. 2003. Tetrahedral mesh generation and optimization based on centroidal Voronoi tessellations. *International Journal for Numerical Methods in Engineering 56*, 9, 1355–1373.

EDELSBRUNNER, H., PREPARATA, F. P., AND WEST, D. B. 1990. Tetrahedrizing point sets in three dimensions. *Journal of Symbolic Computation 10*, 3-4, 335–347.

FREITAG, L. A., AND KNUPP, P. M. 2002. Tetrahedral mesh improvement via optimization of the element condition number. *International Journal for Numerical Methods in Engineering 53*, 6, 1377–1391.

FREITAG, L. A., AND OLLIVIER-GOOCH, C. 1997. Tetrahedral mesh improvement using swapping and smoothing. *International Journal for Numerical Methods in Engineering 40*, 21, 3979–4002.

HUANG, J., CHEN, L., LIU, X., AND BAO, H. 2008. Efficient mesh deformation using tetrahedron control mesh. In *Proceedings of the ACM symposium on Solid and physical modeling*, 241–247.

HUDSON, B., MILLER, G., AND PHILLIPS, T. 2006. Sparse Voronoi refinement. In *Proceedings of the 15th International Meshing Roundtable*, 339–356.

KLINGNER, B. M., AND SHEWCHUK, J. R. 2007. Aggressive tetrahedral mesh improvement. In *Proceedings of the 16th International Meshing Roundtable*, 3–23.

LABELLE, F., AND SHEWCHUK, J. R. 2007. Isosurface stuffing: fast tetrahedral meshes with good dihedral angles. *ACM Trans. Graph. 26*, 3 (July).

LI, X.-Y., TENG, S.-H., AND ÜNGÖR, A. 2000. Biting: Advancing front meets sphere packing. *International Journal for Numerical Methods in Engineering 49*, 1, 61–81.

LIU, A., AND JOE, B. 1995. Quality local refinement of tetrahedral meshes based on 8-subtetrahedron subdivision. *SIAM Journal on Scientific Computing 16*, 6, 1269–1291.

MITCHELL, S. A., AND VAVASIS, S. A. 2000. Quality mesh generation in higher dimensions. *SIAM Journal on Computing 29*, 4, 1334–1370.

MOLINO, N., BRIDSON, R., TERAN, J., AND FEDKIW, R. 2003. A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In *Proceedings of the 12th International Meshing Roundtable*, 103–114.

MOLINO, N., BAO, Z., AND FEDKIW, R. 2004. A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. Graph. 23*, 3 (Aug.), 385–392.

MULLEN, P., MEMARI, P., DE GOES, F., AND DESBRUN, M. 2011. Hot: Hodge-optimized triangulations. *ACM Trans. Graph. 30*, 4 (July), 103:1–103:12.

MÜLLER, M., AND GROSS, M. 2004. Interactive virtual materials. In *Proccedings of Graphics Interface 2004*, 239–246.

MÜLLER, M., TESCHNER, M., AND GROSS, M. 2004. Physically-based simulation of objects represented by surface meshes. In *Proceedings of Computer Graphics International 2004*, 26–33.

NESME, M., KRY, P. G., JEŘÁBKOVÁ, L., AND FAURE, F. 2009. Preserving topology and elasticity for embedded deformable models. *ACM Trans. Graph. 28*, 3 (July), 52:1–52:9.

PARKER, E. G., AND O'BRIEN, J. F. 2009. Real-time deformation and fracture in a game environment. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 156–166.

PERUCCHIO, R., SAXENA, M., AND KELA, A. 1989. Automatic mesh generation from solid models based on recursive spatial decompositions. *International Journal for Numerical Methods in Engineering 28*, 11, 2469–2501.

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

SEDERBERG, T. W., AND PARRY, S. R. 1986. Free-form deformation of solid geometric models. *SIGGRAPH Comput. Graph. 20*, 4 (Aug.), 151–160.

SHEN, C., O'BRIEN, J. F., AND SHEWCHUK, J. R. 2004. Interpolating and approximating implicit surfaces from polygon soup. *ACM Trans. Graph. 23*, 3 (Aug.), 896–904.

SHEPHARD, M. S., AND GEORGES, M. K. 1991. Automatic three-dimensional mesh generation by the finite octree technique. *International Journal for Numerical Methods in Engineering 32*, 4, 709–749.

SHEWCHUK, J. R. 1998. Tetrahedral mesh generation by Delaunay refinement. In *Proceedings of the 14th Annual Symposium on Computational Geometry*, 86–95.

SHEWCHUK, J. R. 2002. What is a good linear element? Interpolation, conditioning, and quality measures. In *Proceedings of the 11th International Meshing Roundtable*, 115–126.

SI, H. 2004. 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.

SIFAKIS, E., SHINAR, T., IRVING, G., AND FEDKIW, R. 2007. Hybrid simulation of deformable solids. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation*, 81–90.

TERAN, J., MOLINO, N., FEDKIW, R., AND BRIDSON, R. 2005. Adaptive physics based tetrahedral mesh generation using level sets. *Engineering with Computers 21*, 1, 2–18.

TOURNOIS, J., SRINIVASAN, R., AND ALLIEZ, P. 2009. Perturbing slivers in 3D Delaunay meshes. In *Proceedings of the 18th International Meshing Roundtable*, 157–173.

TOURNOIS, J., WORMSER, C., ALLIEZ, P., AND DESBRUN, M. 2009. Interleaving delaunay refinement and optimization for practical isotropic tetrahedron mesh generation. *ACM Trans. Graph. 28*, 3 (July), 75:1–75:9.

WOJTAN, C., AND TURK, G. 2008. Fast viscoelastic behavior with thin features. *ACM Trans. Graph. 27*, 3 (Aug.), 47:1–47:8.