A CRYSTALLINE, RED GREEN STRATEGY FOR MESHING HIGHLY DEFORMABLE OBJECTS WITH TETRAHEDRA

Neil Molino[1] Robert Bridson[1] Joseph Teran[1] Ronald Fedkiw[2]

[1]*Stanford University, Stanford, CA, U.S.A.* {*npmolino, rbridson, jteran*}*@stanford.edu*
[2]*Stanford University, Stanford, CA, U.S.A. fedkiw@cs.stanford.edu*

ABSTRACT

Motivated by Lagrangian simulation of elastic deformation, we propose a new tetrahedral mesh generation algorithm that produces both high quality elements *and* a mesh that is well conditioned for subsequent large deformations. We use a signed distance function defined on a Cartesian grid in order to represent the object geometry. After tiling space with a uniform lattice based on crystallography, we use the signed distance function or other user defined criteria to guide a red green mesh subdivision algorithm that results in a candidate mesh with the appropriate level of detail. Then, we carefully select the final topology so that the connectivity is suitable for large deformation and the mesh approximates the desired shape. Finally, we compress the mesh to tightly fit the object boundary using either masses and springs, the finite element method or an optimization approach to relax the positions of the nodes. The resulting mesh is well suited for simulation since it is highly structured, has robust topological connectivity in the face of large deformations, and is readily refined if deemed necessary during subsequent simulation.

Keywords: tetrahedral mesh generation, level set methods, BCC lattice, red green refinement hierarchy, large deformations, muscle simulation

1. INTRODUCTION

We are particularly interested in simulating highly deformable bodies such as the muscle and fatty tissues commonly encountered in biomechanics [1, 2], haptics [3], and virtual surgery [4, 5]. The quality of the tetrahedral mesh has a profound influence on both the accuracy and efficiency of these simulations, see e.g. [5]. Therefore, we propose a mesh generation algorithm designed specifically for such high deformation simulations.

Mesh generation is not only a broad field, but is in some sense many fields, each concerned with the creation of meshes that conform to quality measures specific to the application at hand. *The requirements for fluid flow and heat transfer where the mesh is not deformed, and for small deformation solids where the mesh is barely deformed, can be quite different from those for simulating soft biological tissue that may undergo large deformations.* Simple examples show that the specific requirements or measures of quality of a mesh vary depending on the problem being solved, see e.g. [6].

For example, an optimal mesh for an Eulerian fluid flow simulation should include anisotropically compressed elements in boundary layers, e.g. [7, 8, 9]. In these calculations, the solution gradient in the direction of the fluid flow is typically not as large as in the orthogonal directions. Obviously, it is desirable to have the density of the elements be larger in directions where the gradient is large and lower in directions where the gradient is small, i.e. elongated elements. In contrast, however, highly stretched cells tend to be ill-conditioned when a mesh deforms significantly as is typical for soft bodies. Either the mesh is softer in the thin direction and the cell has a tendency to invert, or

the mesh is stiffer in the thin direction and the simulation becomes very costly since the explicit time step restriction worsens with higher stiffness and smaller element cross-section. Thus, although our method has been designed to provide a high degree of adaptivity both to resolve the geometry and to guarantee quality simulation results, we neither consider nor desire anisotropically stretched elements. Also, since highly deformable bodies tend to be devoid of sharp features such as edges and corners, we do not consider boundary feature preservation.

Our main concern is to generate a mesh that will be robust when subsequently subject to large deformations. For example, although we obviously want an adaptive mesh with smaller elements in areas where more detail is desired, it is even more important to have a mesh that can be adapted during the simulation since these regions will change. Motivated by crystallography, we use a body-centered cubic (BCC) mesh (see e.g. [10]) that is highly structured and produces similar (in the precise geometric sense) tetrahedra under regular refinement. This allows us to adaptively refine both while generating the mesh and during the subsequent simulation.

We start with a uniform tiling of space and use a signed distance function representation of the geometry to guide the creation of the adaptive mesh, the deletion of elements that are not needed to represent the object of interest, and the compression of the mesh necessary to match the object boundaries [11]. This compression stage can be carried out using either a mass spring system, a finite element method or an optimization based approach. One advantage of using a physically based compression algorithm is that it gives an indication of how the mesh is likely to respond to the deformations it will experience during simulation. This is in contrast to many traditional methods that may produce an initial mesh with good quality measures, but also with possible hidden deficiencies that can be revealed during simulation leading to poor accuracy or element collapse. Moreover, our novel topological considerations (discussed below) are specifically designed to address these potential defects present in other mesh generation schemes.

2. RELATED WORK

While Delaunay techniques have been quite successful in two spatial dimensions, they have not been as successful in three spatial dimensions (see e.g. [12] for a discussion of implementation details). They admit flat sliver tetrahedra of negligible volume. Shewchuk provides a nice overview of these methods, including a discussion of why some of the theoretical results are not reassuring in practice [13]. Moreover, he discusses how the worst slivers can often be removed. Cheng et al. [14] also discuss sliver removal, but state that their theorem gives an estimate that is "miserably tiny". Edelsbrunner and Guoy [15] showed that [14] can be used to remove most of the slivers, but is not as promising near boundaries. Another problem with Delaunay methods is that the Delaunay tetrahedralization of a set of points is convex whereas the domains of many finite element calculations are not. Thus, techniques such as the conforming Delaunay approach which inserts additional vertices into the mesh to force it to conform to the boundary of the domain must be developed. The constrained Delaunay tetrahedralization is another method used to enforce boundary recovery [16]. These approaches can be complicated and can even produce an intractably large mesh which is not polynomial in the complexity of the input domain.

Advancing front methods start with a boundary discretization and march a "front" inward forming new elements attached to the existing ones [17]. Advancing front techniques conform well to the boundary. This renders them a useful technique when the specific polygonal boundary representation of the geometry must be matched precisely, for example, when meshing a machine part. When the input geometry is not a polygonal boundary, a triangulation of this boundary must first be performed. The quality of this surface triangulation has a large impact on the three dimensional algorithm's behavior. Poorly shaped surface triangles will engender ill-shaped tetrahedra [18]. A central decision in an advancing front algorithm is the placement of an interior point that marches the front further into the interior of the object. Local element control is possible because new nodes are created at the same time that new elements are created. The node and element creation is done as needed based on local procedures. Authors have experimented with various metrics and criteria to evaluate the placement of the new node, see e.g. [19, 20, 21]. All advancing front techniques have difficulty when fronts merge, however, which unfortunately can occur very near the important boundary in regions of high curvature [8, 9].

Radovitzky and Ortiz [22] started with a face-centered cubic (FCC) lattice defined on an octree and used an advancing front approach to march inward, constructing a mesh with the predetermined nodes of the FCC lattice. They chose FCC over BCC because it gives slightly better tetrahedra for their error bounds. However, after any significant deformation the two meshes will usually have similar character. Moreover, since we keep our BCC connectivity intact (as opposed to [22]), we retain the ability to further refine our BCC mesh during the calculation to obtain locally higher resolution for improved accuracy and robustness. On the other hand, their approach is better at resolving boundary features and is thus likely superior for problems with little to no deformation.

Fuchs [23] begins with a BCC tiling of space which is adaptively refined to obtain the desired nodal density. Vertices outside the object are simply projected to the boundary and then smoothing is applied to optimize the position of the vertices. They emphasize that the BCC connectivity is never used and instead apply Delaunay tesselation. That is, they only use the adaptive BCC lattice to obtain an initial guess for their vertex positions.

Shimada and Gossard [24] packed spheres (or ellipsoids for anisotropic mesh generation [25]) into the domain with mutual attraction and repulsion forces, and generated tetrahedra using the sphere centers as sample points via either a Delaunay or advancing front method. However, *ad hoc* addition and deletion of spheres is required in a search for a steady state, and both local minima and "popping" can be problematic. This led Li et al. [26] to propose the removal of the dynamics from the packing process, instead marching in from the boundary removing spherical "bites" of volume one at a time. This biting was motivated by the advancing front technique, but used here for sphere packing rather than mesh generation. The final mesh is computed with a Delaunay algorithm on the sphere centers. Later, they extended the biting idea to ellipsoids to generate anisotropic meshes [27].

Our compression phase moves the nodes on the boundary of our candidate mesh to the implicit surface, providing boundary conformity. In some sense, this wrapping of our boundary around the level set is related to snakes [28] or GDMs [29] which have been used to triangulate isosurfaces, see e.g. [30]. Neugebauer and Klein started with a marching cubes mesh and moved vertices to the centroid of their neighbors before projecting them onto the zero level set in the neighboring triangles' average normal direction [31]. Grosskopf and Neugebauer improved this method using internodal springs instead of projection to the centroid, incremental projection to the zero isocontour, adaptive subdivision, edge collapse and edge swapping [32]. Kobbelt et al. used related ideas to wrap a mesh with subdivision connectivity around an arbitrary one, but had difficulty projecting nodes in one step, emphasizing the need for slower evolution [33]. To improve robustness, Wood et al. replaced the spring forces with a modified Laplacian smoothing restricted to the tangential direction [34]. Ohtake and Belyaev advocated moving the triangle centroids to the zero isocontour instead of the nodes, and matching the triangle normals with the implicit surface normals [35].

Although we derive motivation from this work, we note that our problem is significantly more difficult since these authors move their mesh in a direction normal to the surface, which is orthogonal to their measure of mesh quality (shapes of triangles tangent to the surface). When we move our mesh normal to the surface, it directly conflicts with the quality of the surface tetrahedra. In [36], de Figueiredo et al. evolved a volumetric mass spring system in order to align it with (but not compress it to) the zero isocontour, but the measure of mesh quality was still perpendicular to the evolution direction since the goal was to triangulate the zero isocontour. Later, however, Velho et al. *did* push in a direction conflicting with mesh quality. They deformed a uniform-resolution Freudenthal lattice to obtain tetrahedralizations using a mass spring model, but were restricted to simple geometries, mostly due to the inability to incorporate adaptivity [37].

In two spatial dimensions, Gloth and Vilsmeier also moved the mesh in a direction that opposed the element quality [38]. They started with a uniform Cartesian grid bisected into triangles, threw out elements that intersected or were outside the domain, and moved nodes to the boundary in the direction of the gradient of the level set function using traditional smoothing, edge swapping, insertion and deletion techniques on the mesh as it deformed.

3. THE BCC LATTICE

We turn our attention to the physical world for inspiration and start our meshing process with a body-centered cubic (BCC) tetrahedral lattice. This mesh has numerous desirable properties and is an actual crystal structure ubiquitous in nature, appearing in vastly different materials such as soft lithium and hard iron crystals, see e.g. [10]. Other spatial tilings are possible. Üngör [39] provides a number of these including tilings using acute tetrahedra.

The BCC lattice consists of nodes at every point of a Cartesian grid along with the cell centers. These node locations may be viewed as belonging to two interlaced grids. Additional edge connections are made between a node and its eight nearest neighbors in the other grid. See figure 1 where these connections are depicted in red and the two interlaced grids are depicted in blue and in green. The BCC lattice is the Delaunay complex of the interlaced grid nodes, and thus possesses all properties of a Delaunay tetrahedralization. Moreover, all the nodes are isomorphic to each other (and in particular have uniform valence), every tetrahedron is congruent to the others, and the mesh is isotropic (so the mesh itself will not erroneously induce any anisotropic bias into a subsequent calculation). The BCC lattice is structured, which may be exploited in preconditioned iterative solvers, multigrid algorithms, etc. and may allow reduced computational and memory requirements.

A significant advantage of the BCC mesh is that it is easily refined initially or during the calculation. Each

regular BCC tetrahedron can be refined into eight tetrahedra, shown in red in figure 2, with a one to eight (or 1:8) refinement. When the shortest of the three possible choices for the edge internal to the tetrahedron is taken, the newly formed tetrahedra are *exactly* the BCC tetrahedra that result from a mesh with cells one half the size. Thus, these eight new tetrahedra are geometrically similar to the tetrahedra of the parent mesh and element quality is guaranteed under this regular 1:8 refinement.

4. A RED GREEN HIERARCHY

Many applications do not require and cannot afford (due to computation time and memory restrictions) a uniformly high resolution mesh. For example, many phenomena such as contact and fracture show highly concentrated stress patterns, often near high surface curvature, outside of which larger tetrahedra are acceptable. In addition, many applications such as virtual surgery can tolerate lower accuracy in the unseen interior of a body. Thus, we require the ability to generate adaptive meshes.

As the BCC lattice is built from cubes, one natural approach to adaptivity is to build its analog based on an octree. We implemented this by adding body centers to the octree leaves, after ensuring the octree was graded with no adjacent cells differing by more than one level. The resulting BCC lattices at different scales were then patched together with special case tetrahedra. For more on octrees in mesh generation, see e.g. [40, 41, 22] (none of which use our multilevel BCC mesh).

However, we found that red green refinement is more economical, simpler to implement, and more flexible, see e.g. [42, 43, 44]. The initial BCC lattice tetrahedra are labelled red, as are any of their eight children obtained with 1:8 subdivision. Performing a red refinement on a tetrahedron creates T-junctions at the newly-created edge midpoints where neighboring

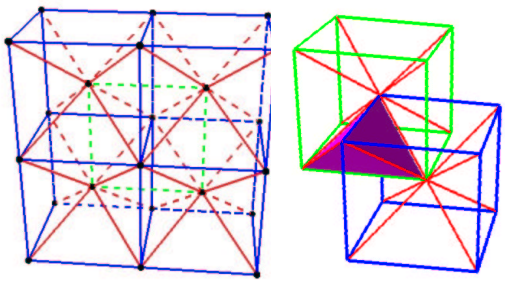

Figure 1: **A portion of the BCC lattice. The blue and the green connections depict the two interlaced grids, and the eight red connections at each node lace these two grids together.**

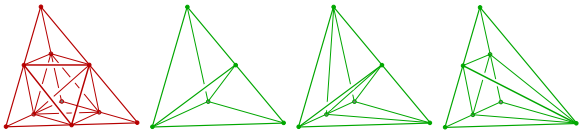

Figure 2: **The standard red refinement (left) produces eight children that reside on a BCC lattice that is one half the size. Three green refinements are allowed (depicted in green).**

tetrahedra are not refined to the same level. To eliminate these, the red tetrahedra with T-junctions are irregularly refined into fewer than eight children by introducing some of the midpoints. These children are labeled green, and are of lower quality than the red tetrahedra that are part of the BCC mesh. Moreover, since they are not BCC tetrahedra, we never refine them. When higher resolution is desired in a region occupied by a green tetrahedron, the entire family of green tetrahedra is removed from its red parent, and the red parent is refined regularly to obtain eight red children that can undergo subsequent refinement.

A red tetrahedron that needs a green refinement can have between one and five midpoints on its edges (in the case of six we do red refinement). We reduce the possibilities for green refinement to those shown in figure 2, adding extra edge midpoints if necessary. This restriction (where all triangles are either bisected or quadrisected) smooths the gradation further and guarantees higher quality green tetrahedra. While there can, of course, be a cascading effect as the extra midpoints may induce more red or green refinements, it is a small price to pay for the superior mesh quality and seems to be a minor issue in practice.

Any criteria may be used to drive refinement, and we experimented with the geometric rules described in the next section. A significant advantage of the red green framework is the possibility for refinement during simulation based on *a posteriori* error estimates, with superior quality guarantees based on the BCC lattice instead of an arbitrary initial mesh. Note that the lower quality green tetrahedra can be replaced by finer red tetrahedra which admit further refinement. However, one difficulty we foresee is in discarding portions of green families near the boundary (see section 6), since part of the red parent is missing. To further refine this tetrahedron, the green family has to be replaced with its red parent which can be regularly refined, then some of the red children need to be discarded and the others must be compressed to the boundary (see sections 7–8). A simpler but lower quality alternative is to arbitrarily relabel those green boundary tetrahedra that are missing siblings as red, allowing them to be directly refined. We plan to address this issue in future work.

5. LEVEL SET GEOMETRY

We represent the geometry with a signed distance function defined on either a uniform grid [45] or an octree grid [46, 47]. In the octree case, we constrain values of fine grid nodes at gradation boundaries to match the coarse grid interpolated values, see e.g. [48]. When the signed distance function has a resolution much higher than that of our desired tetrahedral mesh, we apply motion by mean curvature to smooth the high frequency features and then reinitialize to a signed distance function, see e.g. [45].

Medical data such as the National Library of Medicine's Visible Human data set often comes in the form of volume data [49]. Thus, it is natural to devise a mesh generation technique that generates a volumetric mesh from this data. The data is first converted into a level set using straightforward and efficient algorithms such as a fast marching method [50, 51]. Level sets arise naturally in other applications as well. They are used as a design primitive in CAGD packages. They are also used as a technique to generate a surface from scattered point data [52].

At any point in space, we calculate the distance from the implicitly defined surface as ϕ, which is negative inside and positive outside the surface. To obtain a finer mesh near the boundary, one simply refines tetrahedra that include portions of the interface where $\phi = 0$. If a tetrahedron has nodes with positive values of ϕ and nodes with negative values of ϕ, it obviously contains the interface and can be refined. Otherwise, the tetrahedron is guaranteed not to intersect the interface if the minimum value of $|\phi|$ at a node is larger than the longest edge length (tighter estimates are available, of course). The remaining cases are checked by sampling ϕ appropriately (at the level set grid size $\triangle x$), allowing refinement if any sample is close enough to the interface ($|\phi| < \triangle x$). Figure 3 shows a sphere adaptively refined near its boundary. Note how the interior mesh can still be rather coarse.

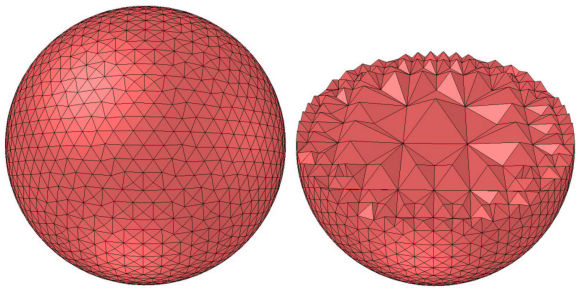

Figure 3: **Tetrahedral mesh of a sphere (18K elements). The cutaway view illustrates that the interior mesh can be fairly coarse even if high resolution is desired on the exterior boundary.**

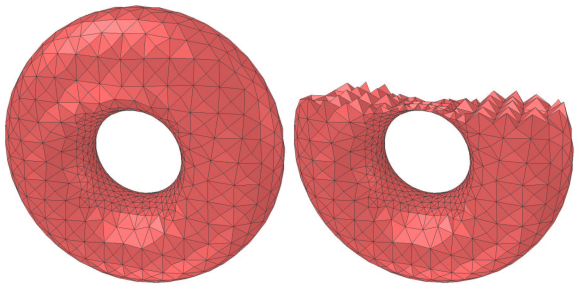

Figure 4: **Tetrahedral mesh of a torus (8.5K elements). Using the principal curvatures increases the level of resolution in the inner ring.**

The outward unit normal is defined as $N = \nabla\phi$ and the mean curvature is defined as $\kappa = \nabla \cdot N$. One may wish to adaptively refine in regions of high curvature, but the mean curvature is a poor measure of this since it is the average of the principal curvatures, $(k_1 + k_2)/2$, and can be small at saddle points where positive and negative curvatures cancel. Instead we use $|k_1| + |k_2|$. The principal curvatures are computed by forming the Hessian, H, and projecting out the components in the normal direction via the projection matrix $P = I - NN^T$. Then the eigenvalues of $PHP/|\nabla\phi|$ are computed, the zero eigenvalue is discarded as corresponding to the eigenvector N, and the remaining two eigenvalues are k_1 and k_2. See e.g. [53]. To detect whether a tetrahedron contains regions of high curvature, we sample at a fine level and check the curvature at each sample point. Figure 4 shows a torus where the inner ring is refined to higher resolution even though the principal curvatures there differ in sign.

6. TOPOLOGICAL CONSIDERATIONS

To obtain the final topology of the mesh, we first cover an appropriately sized bounding box of the object with a coarse BCC mesh. Then we use a conservative discard process to remove tetrahedra that are guaranteed to lie completely outside of the zero isocontour: tetrahedra with four positive ϕ values all larger than the maximum edge length are removed.

In the next step, the remaining tetrahedra are refined according to any user defined criteria, such as indicator variables or geometric properties. We have experimented with using both the magnitude of ϕ and various measures of curvature as discussed in the previous section. Using simply the magnitude of ϕ produces large tetrahedra deep inside the object and a uniform level of refinement around the surface, which can be useful since objects interact with each other via surface tetrahedra. A more sophisticated method uses the surface principal curvatures, better resolving complex geometry and allowing for more robust and efficient simulation when subject to large deformation.

We refine any tetrahedron near the interface if its maximum edge length is too large compared to a radius of curvature measure, $1/(|k_1| + |k_2|)$, indicating an inability to resolve the local geometry. We refine to a user-specified number of levels, resolving T-junctions in the red green framework as needed.

From the adaptively refined lattice we select a subset of tetrahedra that closely matches the object. However, there are specific topological requirements necessary to ensure a valid mesh that behaves well under deformation: the boundary must be a manifold; no tetrahedron may have all four nodes on the boundary; and no interior edge may connect two boundary nodes. Boundary forces can readily crush tetrahedra with all nodes on the boundary, or that are trapped between the boundary and an interior edge with both endpoints on the boundary. To satisfy the conditions, we select all the tetrahedra incident on a set of "enveloped" nodes sufficiently interior to the zero isocontour. This guarantees that every tetrahedron is incident on at least one interior node, and also tends to avoid the bad interior segments for reasonably convex regions, i.e. regions where the geometry is adequately resolved by the nodal samples. We specifically choose the set of nodes where $\phi < 0$ that have all their incident edges at least 25% inside the zero isocontour as determined by linear interpolation of ϕ along the edge.

Additional processing is used to guarantee appropriate topology even in regions where the mesh may be under-resolved. Any remaining interior edges and all edges incident on non-manifold nodes are bisected, and the red green procedure is used to remove all T-junctions. If any refinement is necessary, we recalculate the set of enveloped nodes and their incident tetrahedra as above. As an option, we may add any boundary node with surface degree three to the set of enveloped nodes (if these nodes were to remain, the final surface mesh would typically contain angles over 120°). We also add any non-manifold node that remains and the deeper of the two boundary nodes connected by a bad interior edge. We check that these additions do not create more problems, continuing to add boundary nodes to the set of enveloped nodes until we have achieved all requirements. This quickly and effectively results in a mesh that approximates the object fairly closely (from the viewpoint of an initial guess for the compression phase of the algorithm) and that has connectivity well suited for large deformation simulations.

7. PHYSICS BASED COMPRESSION

We outfit our candidate mesh with a deformable model based on either masses and springs or the finite element method, and subsequently compress the boundary nodes to conform to the zero isocontour of the signed distance function. The compression is driven using either a force or velocity boundary condition on the surface nodes. Applying forces is more robust as it allows the interior mesh to push back, resisting excessive compression while it seeks an optimal state. However, if the internal resistance of the mesh becomes larger than the boundary forces, the boundary will not be matched exactly. Thus, instead of adjusting forces, we switch from force to velocity boundary conditions after an initial stage that carries out most of the needed compression. At each boundary vertex, we choose the direction of the force or constrained velocity component as the average of the incident triangles' normals. No force (or velocity constraint) is applied in other directions so the mesh is free to adjust itself tangentially. The magnitude of the force or velocity constraint is proportional to the signed distance from the level set boundary.

To integrate the equations of motion forward in time, we use a central difference scheme that treats the nonlinear elastic forces explicitly and the damping forces implicitly. This circumvents stringent time step restrictions based on the damping forces. Moreover, since all our damping forces are linear and symmetric negative semi-definite, we can use a conjugate gradient solver for the implicit step. We use a velocity modification procedure to artificially limit the maximum strain of a tetrahedral altitude to 50%, and to artificially limit the strain rate of a tetrahedral altitude to 10% per time step [54]. Since altitudes do not connect two mesh nodes together, all of these operations are carried out by constructing a virtual node at the intersection point between an altitude and the plane containing the base triangle. The velocity of this point is calculated using the barycentric coordinates and velocities of the triangle, and the mass is the sum of the triangle's nodal masses. The resulting impulses on this virtual node are then redistributed to the triangle nodes, conserving momentum.

7.1 Mass Spring Models

The use of springs to aid in mesh generation dates back at least to Gnoffo, who used them to move nodes for two dimensional fluid dynamics calculations [55, 56]. Löhner et al. solved the compressible Euler equations using variable spring stiffnesses to distribute the error evenly over the solution domain [57]. Later, [58] used variational principles analogous to the energy of a system of springs to achieve the same goal. Other authors also measured the error of a CFD calculation along edges of a mesh and then used a spring network to equidistribute these errors over the edges [59, 60, 61]. Bossen and Heckbert point out that internodal forces that both attract and repel (like springs with nonzero rest lengths) are superior to Laplacian smoothing where the nodes only attract each other

[62]. Thus, we use nonzero rest lengths in our springs, i.e. simulating the mesh as if it were a real material. All edges are assigned linear springs obeying Hooke's law, and the nodal masses are calculated by summing one quarter of the mass of each incident tetrahedron.

Edge springs are not sufficient to prevent element collapse. As a tetrahedron gets flatter, the edge springs provide even less resistance to collapse. Various methods to prevent this have been introduced, e.g. [63] proposed a pseudo-pressure term, [64] used an elastic (only, i.e. no damping) force emanating from the barycenter of the tetrahedron. [65] showed that these barycentric springs do not prevent collapse as effectively as altitude springs. In our model, every tetrahedron has four altitude springs each attaching a tetrahedron node to a fictitious node on the plane of its opposite face. Then, the elastic and damping forces are calculated just as for a normal spring. These forces are distributed among the three nodes on the opposite face according to the barycentric weights of the fictitious node. This model has damping forces that are linear and symmetric negative semi-definite in the nodal velocities allowing the damping terms to be integrated using a fast conjugate gradient solver for implicit integration.

When simulating a deformable object with a mass spring network, the material behavior should be independent of mesh refinement. The frequency of a spring scales as $\sqrt{k/ml_o}$ (note our "spring constant" is k/l_o), so the sound speed scales as $l_o\sqrt{k/ml_o} = \sqrt{kl_o/m}$. Requiring the sound speed to be a material property implies that k must scale as m/l_o. Thus, we set the spring stiffness for an edge spring using the harmonic average of the masses of the two nodes at the ends of the spring and its restlength. Similarly, for altitude springs we use the harmonic average of the nodal mass and the triangle mass.

7.2 Finite Element Method

While any number of constitutive models could be used, an interesting strategy is to use the real constitutive model of the material when generating its mesh. In this sense, one might hope to predict how well the mesh will react to subsequent deformation during simulation, and possibly work to ensure simulation robustness while constructing the mesh.

We use the nonlinear Green strain tensor, $G = 1/2[(\partial x/\partial u)^T(\partial x/\partial u) - I]$, where $x(u)$ represents a point's position in world coordinates as a function of its coordinates in object space. Isotropic, linearly-elastic materials have a stress strain relationship of the form $S_e = \lambda tr(G)I + 2\mu G$ where λ and μ are the Lamé coefficients. Damping stress is modeled similarly with $S_d = \alpha tr(\nu)I + 2\beta\nu$, where $\nu = \partial G/\partial t$ is the strain rate. The total stress tensor is then $S = S_e + S_d$.

We use linear basis functions in each tetrahedron so that the displacement of material is a linear function of the tetrahedron's four nodes. From the nodal locations and velocities we obtain this linear mapping and its derivative and use them to compute the strain and the strain rate, which in turn are used to compute the stress tensor. Finally, because the stress tensor encodes the force distribution inside the material, we can use it to calculate the force on the nodes.

In their finite element simulation, [66] added a force in the same direction as our altitude springs. Since that force was the same on all nodes and based on the volume deviation from the rest state, it does not adversely penalize overly compressed directions and can even exacerbate the collapse. Instead, we artificially damp the strain and strain rate of the altitudes of the tetrahedra as discussed above.

8. OPTIMIZATION BASED COMPRESSION

As an alternative to physical simulation, one can directly optimize mesh quality metrics such as aspect ratios. This does not provide the same feedback on potential problems for subsequent simulation, but can give better quality measures since they are directly pursued with each movement of a node. Coupled with our robust connectivity (see section 6), this produces excellent results. Freitag and Ollivier-Gooch [67] demonstrated that optimizing node positions in a smoothing sweep, i.e. placing one node at a time at a location that maximizes the quality of incident elements, is superior to Laplacian smoothing in three spatial dimensions. We combine this optimization sweeping with boundary constraints by first moving boundary nodes in the incident triangles' average normal direction by an amount proportional to the local signed distance value. Then the optimization is constrained to only move boundary nodes in the tangential direction.

It is important to move boundary nodes gradually over several sweeps just as with physical simulation, since otherwise the optimization gets stuck in local extrema. We also found it helpful to order the nodes in the sweep with the boundary nodes first, their interior neighbors next, and so on into the interior. Then we sweep in the reverse order and repeat. This efficiently transfers information from the boundary compression to the rest of the mesh. Typically, we do five sweeps of moving the boundary nodes 1/3 of the signed distance in the mesh normal direction, then finish off with five to ten sweeps moving boundary nodes the full signed distance to ensure a tight boundary fit. To speed up the sweeps, we do not bother moving nodes that are incident on tetrahedra of sufficiently high quality relative to the worst tetrahedron currently in the mesh. In the initial

sweeps we end up only optimizing roughly 10% of the nodes, and in the final sweeps we optimize 30%-50% of the nodes.

While more efficient gradient methods may be used for the nodal optimization, we found a simple pattern search (see e.g. [68]) to be attractive for its robustness, simplicity of implementation, and flexibility in easily accommodating any quality metric. For interior nodes we used seven well spread-out directions in the pattern search. We implemented the normal direction constraint on boundary nodes simply by choosing five equally spaced pattern directions orthogonal to the average mesh normal at the node. The initial step size of the pattern search was .05 times the minimum distance to the opposite triangle in any tetrahedron incident on the node (to avoid wasting time on steps that crush elements). After four "strikes" (searches at a given step size that yielded no improvement in quality, causing the step size to be halved) we move to the next node. For interior nodes we use as a quality metric the minimum of $\frac{a}{L} + \frac{1}{4}\cos(\theta_M)$ over the incident tetrahedra, where a is the minimum altitude length, L is the maximum edge length, and θ_M is the maximum angle between face normals. For surface nodes we add to this a measure of the quality of the incident boundary triangles, the minimum of $\frac{a_t}{L_t} + \frac{1}{\psi_M}$ where a_t is the minimum triangle altitude, L_t is the maximum triangle edge, and ψ_M is the maximum triangle angle. We found that including the extra terms beyond the tetrahedron aspect ratios helped guide the optimization out of local minima and actually resulted in better aspect ratios.

9. RESULTS

We demonstrate several examples of tetrahedral meshes that were generated with our algorithm. The results for all three compression techniques are comparable, with the FEM simulations taking slightly longer

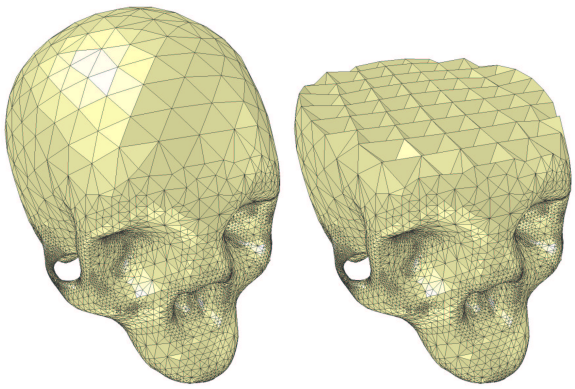

Figure 5: **Tetrahedral mesh (left) and cutaway view (right) of a cranium (80K elements).**

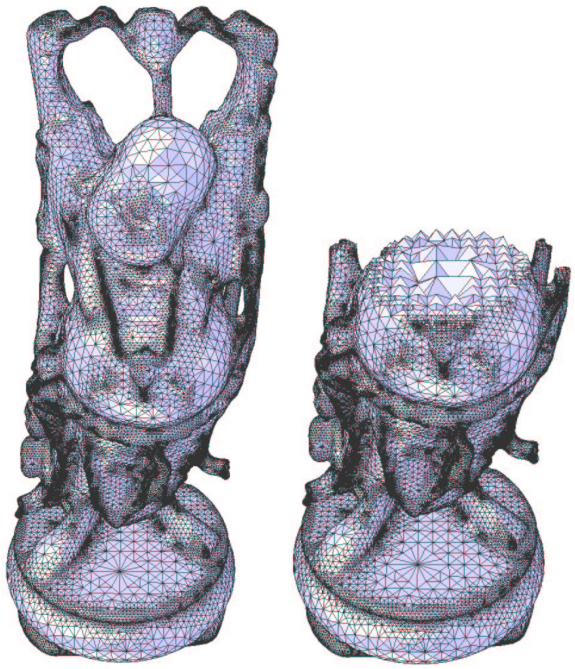

Figure 6: **Tetrahedral mesh (left) and cutaway view (right) of a model Buddha (800K elements).**

(ranging from a few minutes to a few hours on the largest meshes) than the mass spring methods, but producing a slightly higher quality mesh. For example, the maximum aspect ratio of a tetrahedron in the cranium generated with finite elements is 6.5, whereas the same mesh has a maximum aspect ratio of 6.6 when the final compression is done using a mass spring model. Mass spring networks have a long tradition in mesh generation, but a finite element approach offers greater flexibility and robustness that we anticipate will allow better three-dimensional mesh generation in the future. Currently the fastest method is the optimization based compression, roughly faster by a factor of ten.

We track a number of quality measures including the maximum aspect ratio (defined as the tetrahedron's maximum edge length divided by its minimum altitude), minimum dihedral angle, and maximum dihedral angle during the compression phase. The maximum aspect ratios of our candidate mesh start at about 3.5 regardless of the degree of adaptivity, emphasizing the desirability of our combined red green adaptive BCC approach. This number comes from the green tetrahedra (the red tetrahedra have aspect ratios of $\sqrt{2}$). In the more complicated models, the worst aspect ratio in the mesh tends to increase to around 6–8 for the physics based compression methods and to around 5–6 for the optimization based compression.

For the cranium model, the physics based compression methods gave a maximum aspect ratio of 6.5 and aver-

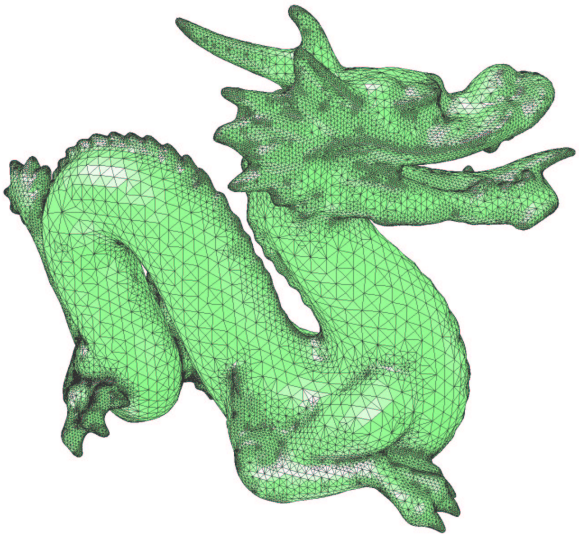

Figure 7: **Tetrahedral mesh of a model dragon (500K elements).**

age aspect ratio of 2.1, with dihedral angles bounded between 17° and 147°. The dragon mesh has a maximum aspect ratio of 7.6 and an average aspect ratio of 2.2, with dihedral angles bounded between 13° and 154°. The buddha model was more challenging, giving a worst aspect ratio of 8.1 and average of 2.3, and dihedral angles between 13° and 156°. Using optimization on the same examples yielded better results, listed in table 1, where we have also listed a measure of adaptivity, the ratio of the longest edge in the mesh to the shortest. The aspect ratios all drop below 6, i.e. less than twice the initial values.

Of course, these results are dependent on the types and strengths of springs, the constitutive model used in the FEM, and the quality measures used in the optimization based technique. It is easier to achieve good quality with the optimization technique since one simply optimizes based on the desired measure, as opposed to the physics based techniques where one has to choose parameters that indirectly lead to a quality mesh. However, we stress that the measure of mesh quality *is* the measure of the worst element at any point of dynamic simulation. It does little good to have a perfect mesh that collapses immediately when the simulation begins. For meshes that undergo little to no deformation (fluid flow, heat flow, small strain, etc.) this quality measure is either identical to or very close to that of the initial mesh. However, for large deformation problems this is not the case, and the physics based compression techniques hold promise in the sense that the resulting mesh may be better conditioned for simulation. We believe an interesting possibility for the future would be to consider hybrid approaches that use the physics based compression algorithms to guide an optimization procedure to avoid local minima.

Example	Cranium	Dragon	Buddha
max aspect ratio	4.5	5.3	5.9
avg aspect ratio	2.3	2.3	2.3
min dihedral	18°	16°	16°
max dihedral	145°	150°	150°
max/min edge	94	94	100

Table 1: **Quality measures for the optimization example meshes. The aspect ratio is defined as the longest edge over the shortest altitude. The max/min edge length ratio indicates the degree of adaptivity.**

10. EXAMPLE: MUSCLE SIMULATION

Musculoskeletal simulation is an active research area in biomechanics. We demonstrate the robustness of our meshing algorithm by simulating volumetric, deformable skeletal muscle. Our meshing algorithm allows us to create high resolution muscle, tendon and bone geometries from the Visible Human data set [49]. The data for these biological materials are originally in the form of a segmented series of consecutive images that can be used to create a level set description of each tissue geometry. This level set can then be used with either the dynamic or optimization based algorithm. Figure 8 shows an adaptive resolution biceps with tendon that was created using dynamic meshing with a finite element constitutive model.

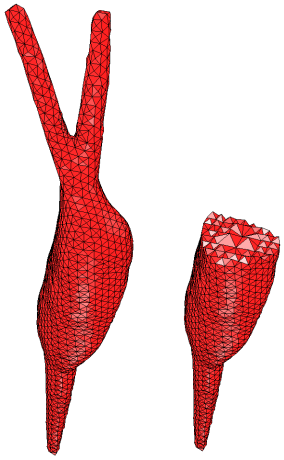

Figure 8: **Adaptive resolution mesh of the right biceps with proximal and distal tendons.**

We simulate both isotonic and isometric contraction of the right biceps and triceps with a state-of-the-art biomechanical model for hyperelastic material response, neurological activation level and fiber anatomy. Muscle is a fibrous structure composed of fascicles embedded in a matrix of isotropic material [69], and we use a nonlinear transversely-isotropic quasi-incompressible constitutive model [70, 71] to represent this structure during simulation. The hyperelastic strain energy associated with this model is a sum of three terms: the first term represents the incompressibility of biological tissues and penalizes volume

change; the second term represents the isotropic embedding matrix; and the third term is the transversely-isotropic component that models muscle fiber contraction and is based on the standard muscle force/length curve [72]. This model can be used in both muscle and tendon, however, tendon tends to be as much as an order of magnitude stiffer and muscle has an additional contractile force added to the fiber component that depends on the muscle activation level.

In addition to activation level, muscle (and tendon) models need information about the local fiber direction. Muscle fiber arrangements vary in complexity from being relatively parallel and uniform to exhibiting several distinct regions of fiber directions. We use a B-spline solid as in [73, 74] to represent more intricate muscle fiber architectures and to assign a fiber direction to individual tetrahedra in the mesh. During both isometric and isotonic contraction, muscles are given a varying activation level throughout the simulation. The activation levels are computed from key-frames of the skeletal animation, using an established biomechanics analysis known as muscle force distribution [75] to compute activations of redundant sets of muscles.

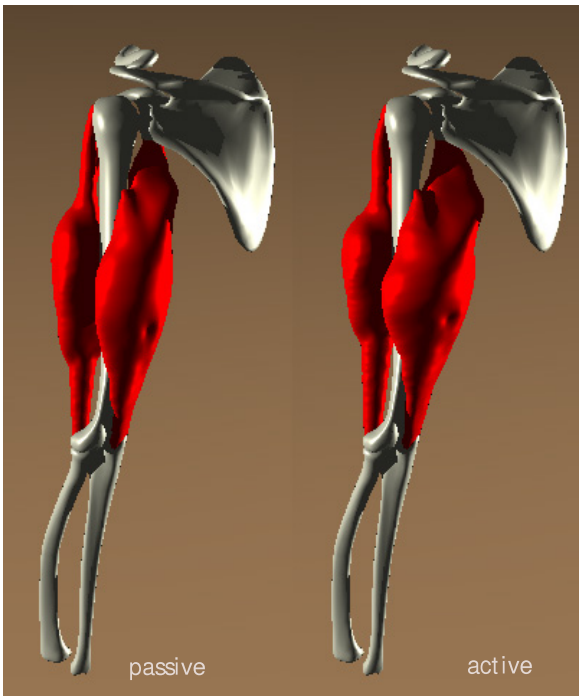

Figure 9: **Simulation of isometric contraction. A posterior (from behind) view of the upper arm shows contraction of the triceps muscle and the partially occluded biceps muscle from passive (left) to full activation (right).**

Figures 9 and 10 show sample frames of our musculoskeletal simulations. Figure 9 depicts relaxed and active muscle during isometric contraction. In this simulation the activation level in the two muscles increases from 0 (fully relaxed) to 1 (fully activated) and back to 0 over the span of two seconds. The bulging in the bellies of the muscles results from larger stiffness in the tendons. Figure 10 shows several frames of musculoskeletal motion. The motion of the kinematic skeleton was key-framed (although our framework allows for motion data from other sources like motion capture). At each key-frame in the animation, an inverse dynamics analysis was computed for the biceps and triceps activation levels required to maintain the static pose. These activation levels were then interpolated in time and used for the dynamic muscle simulation.

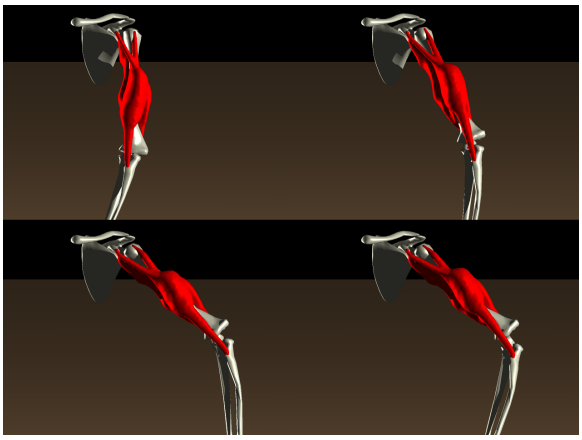

Figure 10: **Muscle contraction with skeletal motion.**

Figure 11 shows the relative change in maximum aspect ratio observed during an isometric contraction of the biceps for meshes created using the optimization algorithm and using the dynamics algorithm. Similar results were observed for the triceps and during isotonic contraction. These results suggest that initial mesh quality may be misleading and not sufficient to guarantee performance of a mesh throughout simulation. In all of our comparisons, the optimization based meshes were of higher quality initially, but tended to undergo as much as a 70% change in maximum aspect ratio during muscle contraction, whereas the dynamics based meshes tended to degrade by only 25%. Of course, if the initial optimization mesh is of significantly higher quality then the overall maximum aspect ratio will still be lower. We are not yet claiming that a particular method is better, but simply pointing out that the initial mesh quality is not always a reliable predictor of performance during subsequent simulation.

11. CONCLUSIONS

We presented an algorithm for producing a high quality tetrahedral mesh directly from a level set. The focus of this algorithm is the generation of a tetrahe-

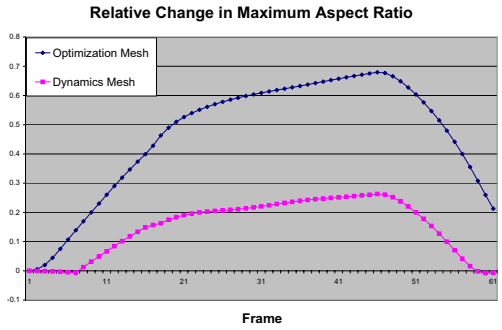

Figure 11: **Plot of changes in maximum aspect ratios during simulation of isometric contraction for dynamics and optimization based meshes.**

dral mesh designed specifically for high deformation. Key points of our algorithm that make it particularly well suited for high deformation are: the use of a red green strategy in conjunction with a BCC lattice making the usually temperamental red green approach robust and suitable for subsequent simulation (and enhancing multiresolution capabilities); the identification and avoidance of connectivity that is problematic for large deformations in constructing the mesh; and the use of simulation and constitutive models to generate the mesh, thus identifying potential weaknesses before simulation even begins (in fact this is what originally led us to the problematic connectivity). Finally, we simulated a few muscles from the NIH Visible Human data set to demonstrate the efficacy of these meshes. In particular, we illustrated that although initial mesh quality measures are important, they do not guarantee high quality during subsequent simulation, and can in fact be misleading.

12. ACKNOWLEDGEMENTS

Research supported in part by an ONR YIP award and PECASE award (ONR N00014-01-1-0620), a Packard Foundation Fellowship, a Sloan Research Fellowship, ONR N00014-03-1-0071, ONR N00014-02-1-0720, NSF ITR-0121288, NSF ACI-0205671 and NSF DMS-0106694. In addition, N. M. and R. B. were supported in part by a Stanford Graduate Fellowships, and J. T. was supported in part by an NSF Graduate Research Fellowship.

References

[1] Martins J., Pires E., Salvado R., Dinis P. "A Numerical Model of Passive and Active Behavior of Skeletal Muscles." *Comput. Meth. in Appl. Mech. and Eng.*, vol. 151, 419–433, 1998

[2] Hirota G., Fisher S., State A., Lee C., Fuchs H. "An Implicit Finite Element Method for Elastic Solids in Contact." *Comput. Anim.* 2001

[3] Cotin S., Delingette H., Ayache N. "Real-Time Volumetric Deformable Models for Surgery Simulation." *Proc. of Vis. in Biomed. Comput.*, pp. 535–540. 1996

[4] Ganovelli F., Cignoni P., Montani C., Scopigno R. "A Multiresolution Model for Soft Objects Supporting Interactive Cuts and Lacerations." *Eurographics*, pp. 271–282. 2000

[5] Bro-Nielsen M., Cotin S. "Real-time Volumetric Deformable Models for Surgery Simulation using Finite Elements and Condensation." *Comput. Graph. Forum*, vol. 15, no. 3, 57–66, August 1996

[6] Fleishmann P., Kosik R., Selberherr S. "Simple Mesh Examples to Illustrate Specific Finite Element Mesh Requirements." *8th Int. Meshing Roundtable*, pp. 241–246. 1999

[7] Marcum D.L. "Generation of Unstructured Grids for Viscous Flow Applications." *AIAA*. 1995

[8] Garimella R., Shephard M. "Boundary Layer Meshing for Viscous Flows in Complex Domains." *7th Int. Meshing Roundtable*, pp. 107–118. 1998

[9] Lohner R., Cebral J. "Generation of Non-Isotropic Unstructured Grids via Directional Enrichment." *2nd Symp. on Trends in Unstructured Mesh Generation.* 1999

[10] Burns G., Glazer A.M. *Space Groups for Solid State Scientists.* Academic Press, Inc., 1990. (2nd ed.)

[11] Osher S., Sethian J. "Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations." *J. Comp. Phys.*, vol. 79, 12–49, 1988

[12] Weatherill N.P., Hassan O. "Efficient Three-Dimensional Delaunay Triangulation with Automatic Point Creation and Imposed Boundary Constraints." *Int. J. for Num. Meth. in Eng.*, vol. 37, 2005–2039, 1994

[13] Shewchuk J. "Tetrahedral Mesh Generation by Delaunay Refinement." *Proc. 14th Annual Symp. on Comput. Geom.*, pp. 86–95. 1998

[14] Cheng S.W., Dey T.K., Edelsbrunner H., Facello M.A., Teng S.H. "Sliver Exudation." *J. of the ACM*, vol. 47, no. 5, 883–904, 2000

[15] Edelsbrunner H., Guoy D. "An Experimental Study of Sliver Exudation." *Eng. Comput.*, vol. 18, no. 3, 229–240, 2002

[16] Shewchuk J. "Constrained Delaunay Tetrahedralizations and Provably Good Boundary Recovery." *11th Int. Meshing Roundtable.* 2002

[17] Schöberl J. "NETGEN - An Advancing Front 2D/3D Mesh Generator Based on Abstract Rules." *Comput. and Vis. in Science*, vol. 1, 41–52, 1997

[18] Möller P., Hansbo P. "On Advancing Front Mesh Generation in Three Dimensions." *Int. J. for Num. Meth. in Eng.*, vol. 38, 3551–3569, 1995

[19] Lo S.H. "Volume Discretization into Tetrahedra - I., Verification and Orientation of Boundary Surfaces." *Comput. and Structures*, vol. 39, no. 5, 493–500, 1991

[20] Lo S.H. "Volume Discretization into Tetrahedra - II., 3D Triangulation by Advancing Front Approach." *Comput. and Structures*, vol. 39, no. 5, 501–511, 1991

[21] Mavriplis D.J. "An Advancing Front Delaunay Triangulation Algorithm Designed for Robustness." *J. Comp. Phys.*, vol. 117, 90–101, 1995

[22] Radovitzky R.A., Ortiz M. "Tetrahedral Mesh Generation Based in Node Insertion in Crystal Lattice Arrangements and Advancing-Front Delaunay Triangulation." *Comput. Meth. in Appl. Mech. and Eng.*, vol. 187, 543–569, 2000

[23] Fuchs A. "Automatic Grid Generation with Almost Regular Delaunay Tetrahedra." *7th Int. Meshing Roundtable*, pp. 133–148. 1998

[24] Shimada K., Gossard D. "Bubble Mesh: Automated Triangular Meshing of Non-Manifold Geometry by Sphere Packing." *ACM 3rd Symp. on Solid Model. and Appl.*, pp. 409–419. 1995

[25] Yamakawa S., Shimada K. "High Quality Anisotropic Tetrahedral Mesh Generation via Packing Ellipsoidal Bubbles." *9th Int. Meshing Roundtable*, pp. 263–273. 2000

[26] Li X., Teng S., Üngör A. "Biting Spheres in 3D." *8th Int. Meshing Roundtable*, pp. 85–95. 1999

[27] Li X., Teng S., Üngör A. "Biting Ellipses to Generate Anisotropic Mesh." *8th Int. Meshing Roundtable*, pp. 97–108. 1999

[28] Kass M., Witkin A., Terzopoulos D. "Snakes: Active Contour Models." *Int. J. of Comput. Vis.*, pp. 321–331.

1987

[29] Miller J., Breen D., Lorensen W., O'Bara R., Wozny M. "Geometrically Deformed Models: A Method for Extracting Closed Geometric Models from Volume Data." *Comput. Graph. (SIGGRAPH Proc.)*, pp. 217–226, 1991

[30] Sadarjoen I.A., Post F.H. "Deformable Surface Techniques for Field Visualization." *Eurographics*, pp. 109–116. 1997

[31] Neugebauer P., Klein K. "Adaptive Triangulation of Objects Reconstructed from Multiple Range Images." *Vis.* 1997

[32] Grosskopf S., Neugebauer P.J. "Fitting Geometrical Deformable Models to Registered Range Images." *European Wrkshp. on 3D Structure from Multiple Images of Large-Scale Environments (SMILE)*, pp. 266–274. 1998

[33] Kobbelt L.P., Vorsatz J., Labsik U., Seidel H.P. "A Shrink Wrapping Approach to Remeshing Polygonal Surfaces." *Eurographics*, pp. 119–130. 1999

[34] Wood Z., Desbrun M., Schröder P., Breen D. "Semi-Regular Mesh Extraction from Volumes." *Vis.*, pp. 275–282. 2000

[35] Ohtake Y., Belyaev A.G. "Dual/Primal Mesh Optimization for Polygonized Implicit Surfaces." *Proc. of the 7th ACM Symp. on Solid Model. and Appl.*, pp. 171–178. ACM Press, 2002

[36] de Figueiredo L.H., Gomes J., Terzopoulos D., Velho L. "Physically-Based Methods for Polygonization of Implicit Surfaces." *Proc. of the Conf. on Graph. Interface*, pp. 250–257. 1992

[37] Velho L., Gomes J., Terzopoulos D. "Implicit Manifolds, Triangulations and Dynamics." *J. of Neural, Parallel and Scientific Comput.*, vol. 15, no. 1–2, 103–120, 1997

[38] Gloth O., Vilsmeier R. "Level Sets as Input for Hybrid Mesh Generation." *9th Int. Meshing Roundtable*, pp. 137–146. 2000

[39] Üngör A. "Tiling 3D Euclidean Space with Acute Tetrahedra." *Proc. of the Canadian Conf. on Comp. Geom.*, pp. 169–172. 2001

[40] Yerry M.A., Shephard M.S. "Automatic Three-Dimensional Mesh Generation By The Modified Octree Technique." *Int. J. For Num. Meth. in Eng.*, vol. 20, 1965–1990, 1984

[41] Shephard M.S., Georges M.K. "Automatic Three-Dimensional Mesh Generation by the Finite Octree Technique." *Int. J. for Num. Meth. Eng.*, vol. 32, 709–739, 1991

[42] Bey J. "Tetrahedral Grid Refinement." *Computing*, vol. 55, 355–378, 1995

[43] Grosso R., Lürig C., Ertl T. "The Multilevel Finite Element Method for Adaptive Mesh Optimization and Visualization of Volume Data." *Visualization*, pp. 387–394. 1997

[44] de Cougny H.L., Shephard M.S. "Parallel Refinement and Coarsening of Tetrahedral Meshes." *Int. J. for Num. Meth. in Eng.*, vol. 46, 1101–1125, 1999

[45] Osher S., Fedkiw R. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, 2002. New York, NY

[46] Strain J. "Fast Tree-Based Redistancing for Level Set Computations." *J. Comput. Phys.*, vol. 152, 664–686, 1999

[47] Strain J. "Tree Methods for Moving Interfaces." *J. Comput. Phys.*, vol. 151, 616–648, 1999

[48] Westermann R., Kobbelt L., Ertl T. "Real-Time Exploration of Regular Volume Data by Adaptive Reconstruction of Isosurfaces." *The Vis. Comput.*, vol. 15, no. 2, 100–111, 1999

[49] U.S. National Library of Medicine. "The Visible Human Project.", 1994. Http://www.nlm.nih.gov/research/visible/

[50] Tsitsiklis J. "Efficient Algorithms for Globally Optimal Trajectories." *IEEE Trans. on Automatic Control*, vol. 40, 1528–1538, 1995

[51] Sethian J. "A Fast Marching Level Set Method for Monotonically Advancing Fronts." *Proc. Natl. Acad. Sci.*, vol. 93, 1591–1595, 1996

[52] Zhao H.K., Osher S., Fedkiw R. "Fast Surface Reconstruction Using the Level Set Method." *1st IEEE Wrkshp. on Variational and Level Set Meth., 8th Int. Conf. on Comput. Vis.*, pp. 194–202. 2001

[53] Ambrosio L., Soner H.M. "Level Set Approach to Mean Curvature Flow in Arbitrary Codimension." *J. of Differential Geometry*, vol. 43, 693–737, 1996

[54] Bridson R., Fedkiw R., Anderson J. "Robust Treatment of Collisions, Contact and Friction for Cloth Animation." *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 21, 594–603, 2002

[55] Gnoffo P. "A Vectorized, Finite-Volume, Adaptive-Grid Algorithm for Navier-Stokes Calculations." *Num. Grid Generation*, pp. 819–835, 1982

[56] Gnoffo P. "A Finite-Volume, Adaptive Grid Algorithm Applied to Planetary Entry Flowfields." *AIAA*. 1982

[57] Lohner R., Morgan K., Zienkiewicz O.C. *Adaptive Grid Refinement for Compressible Euler Equations*, pp. 281–297. John Wiley and Sons Ltd., 1986

[58] Nakahashi K., Deiwert G.S. "Self-Adaptive-Grid Method with Application to Airfoil Flow." *AIAA*, vol. 25, no. 4, 513–520, 1987

[59] Dompierre J., Vallet M., Fortin M., Habashi W.G., Aït-Ali-Yahia D., Boivin S., Bourgault Y., Tam A. "Edge-Based Mesh Adaptation for CFD." *Conf. on Num. Meth. for the Euler and Navier-Stokes Equations*. 1995

[60] Vallet M., Dompierre J., Bourgault Y., Fortin M., Habashi W.G. "Coupling Flow Solvers and Grids through an Edge-Based Adaptive Grid Method." *Fluids Eng. Division Conf.*, vol. 3. 1996

[61] Fortin M., Vallet M., Dompierre J., Bourgault Y., Habashi W.G. "Anisotropic Mesh Adaptation: Theory, Validation, and Applications." *Comput. Fluid Dynamics*. 1996

[62] Bossen F.J., Heckbert P.S. "A Pliant Method for Anisotropic Mesh Generation." *5th Int. Meshing Roundtable*, pp. 63 – 76. 1996

[63] Palmerio B. "An Attraction-Repulsion Mesh Adaption Model for Flow Solution on Unstructured Grids." *Comput. and Fluids*, vol. 23, no. 3, 487–506, 1994

[64] Bourguignon D., Cani M.P. "Controlling Anisotropy in Mass-Spring Systems." *Eurographics*, pp. 113–123. 2000

[65] Cooper L., Maddock S. "Preventing Collapse Within Mass-Spring-Damper Models of Deformable Objects." *The 5th Int. Conf. in Central Europe on Comput. Graphics and Vis.* 1997

[66] Picinbono G., Delingette H., Ayache N. "Non-Linear and Anisotropic Elastic Soft Tissue Models for Medical Simulation." *IEEE Int. Conf. Robot. and Automation*. 2001

[67] Freitag L., Ollivier-Gooch C. "Tetrahedral Mesh Improvement Using Swapping and Smoothing." *Int. J. for Num. Meth. in Eng.*, vol. 40, 3979–4002, 1997

[68] Torczon V. "On the Convergence of Pattern Search Algorithms." *SIAM J. Opt.*, vol. 7, no. 1, 1–25, 1997

[69] Fung Y.C. *Biomechanics: Mechanical Properties of Living Tissues*. Springer-Verlag, New York, 1981

[70] Yucesoy C.A., Koopman B.H., Huijing P.A., Grootenboer H.J. "Three-Dimensional Finite Element Modeling of Skeletal Muscle Using a Two-Domain Approach: Linked Fiber-Matrix Mesh Model." *J. of Biomech.*, vol. 35, 1253–1262, 2002

[71] Weiss J., Maker B., Govindjee S. "Finite-element impementation of incompressible, transversely isotropic hyperelasticity." *Comput. Meth. in Appl. Mech. and Eng.*, vol. 135, 107–128, 1996

[72] Zajac F. "Muscle and Tendon: Properties, Models, Scaling, and Application to Biomechanics and Motor Control." *Critical Reviews in Biomed. Eng.*, vol. 17, no. 4, 359–411, 1989

[73] Ng-Thow-Hing V., Fiume E. "Interactive Display and Animation of B-Spline Solids as Muscle Shape Primitives." D. Thalmann, M. van de Panne, editors, *Proc. of the Eurographics Wrkshp. on Comput. Anim. and Sim.* Springer Verlag, 1997

[74] Ng-Thow-Hing V., Fiume E. "Application-Specific Muscle Representations." W. Sturzlinger, M. McCool, editors, *Proc. of Gr. Inter. 2002*, pp. 107–115. Canadian Information Processing Society, 2002

[75] Crowninshield R. "Use of Optimization Techniques to Predict Muscle Forces." *Trans. of the ASME*, vol. 100, 88–92, may 1978