Engineering with Computers (2005) 21: 2–18
DOI 10.1007/s00366-005-0308-8

SPECIAL ISSUE

J. Teran · Neil Molino · R. Fedkiw · R. Bridson

# Adaptive physics based tetrahedral mesh generation using level sets

Received: 14 September 2003 / Accepted: 13 October 2004 / Published online: 12 July 2005


**Abstract** We present a tetrahedral mesh generation algorithm designed for the Lagrangian simulation of deformable bodies. The algorithm's input is a level set (i.e., a signed distance function on a Cartesian grid or octree). First a bounding box of the object is covered with a uniform lattice of subdivision-invariant tetrahedra. The level set is then used to guide a red green adaptive subdivision procedure that is based on both the local curvature and the proximity to the object boundary. The final topology is carefully chosen so that the connectivity is suitable for large deformation and the mesh approximates the desired shape. Finally, this candidate mesh is compressed to match the object boundary. To maintain element quality during this compression phase we relax the positions of the nodes using finite elements, masses and springs, or an optimization procedure. The resulting mesh is well suited for simulation since it is highly structured, has topology chosen specifically for large deformations, and is readily refined if required during subsequent simulation. We then use this algorithm to generate meshes for the simulation of skeletal muscle from level set representations of the anatomy. The geometric complexity of biological materials makes it very difficult to generate these models procedurally and as a result we obtain most if not all data from an actual human subject. Our current method involves using voxelized data from the Visible Male [1] to create level set representations of muscle and bone geometries. Given this representation, we use simple level set operations to rebuild and repair errors in the segmented data as well as to smooth aliasing inherent in the voxelized data.

J. Teran (✉) · N. Molino (✉) · R. Fedkiw (✉)
Stanford University, Stanford, CA, USA
E-mail: jteran@stanford.edu
E-mail: npmolino@stanford.edu
E-mail: fedkiw@cs.stanford.edu

R. Bridson (✉)
University of British Columbia, Vancouver, BC, Canada
E-mail: rbridson@cs.ubc.ca

**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 [2, 3], haptics [4], and virtual surgery [5, 6]. The stability, accuracy, and efficiency of these simulations are all very dependent upon the quality of the tetrahedral mesh, see e.g. [6]. Therefore, our mesh generation algorithm is designed specifically for such large deformation simulations, and even includes simulation as a substantial component.

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 a target application. Fluid flow and heat transfer meshes which are not deformed at all, or small deformation solids meshes which are barely deformed, impose requirements on the mesh that can be quite different from those for simulating soft biological tissue that may undergo large deformations. Fleishman et al. [7] uses simple examples to show that the specific measures of mesh quality vary depending on the problem being solved.

Eulerian fluid flow simulations require anisotropically compressed elements in boundary layers, e.g. [8–10]. In these calculations, the solution gradient is typically smaller in the direction of the fluid flow than it is in orthogonal directions. Obviously, it is desirable to have the density of the elements be higher in directions where the gradient is high and lower in directions where the gradient is low, i.e. elongated elements. In contrast, highly stretched cells tend to be ill-conditioned when a mesh deforms significantly, as is typical for soft bodies. If the mesh is softer in the thin direction, then the cells will have a tendency to invert, which can terminate a

simulation. In contrast, if the material is stiffer in the thin direction, then the calculation will be very expensive because explicit time step restrictions become even more stringent with higher stiffness or with small 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. [11]) 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.

The signed distance function representation of the object's geometry is useful for many reasons. First, it provides a natural bounding box for the object, which we tile with a uniform lattice of tetrahedra. Then, it can be used to guide the creation of an adaptive mesh by providing geometric information anywhere in space. The deletion of elements that are completely outside the object of interest is facilitated by the $O(1)$ inside/outside testing provided by the level set. Finally, the compression phase uses the level set value to guide tetrahedron nodes that remain outside the surface following the discard phase [12]. Nodes that are interior to the surface are treated differently to avoid element collapse.

We take measures to produce meshes well suited for large deformation simulations. The compression stage of our algorithm can be carried out using a mass spring system, a finite element constitutive model or an optimization based approach. One advantage of using a physically based compression algorithm is that it can forecast how the mesh is likely to respond to the deformations it will experience during simulation. This is in contrast to many traditional purely geometric techniques 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. We also specifically avoid pathological connectives which engender mesh defects and collapsing elements.

## 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. [13] 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 [14]. Moreover, he discusses how the worst slivers can often be removed. Cheng et al. [15] also discuss sliver removal, but state that their theorem gives an estimate that is "miserably tiny". Edelsbrunner and Guoy [16] showed that [15] it 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 [17]. 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 [18]. 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 [19]. 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 according to local procedures. Authors have experimented with various metrics and criteria to evaluate the placement of the new node, see e.g. [20–22]. Advancing front techniques have difficulty when fronts merge, however, which unfortunately can occur very near the important boundary in regions of high curvature [9, 10].

Radovitzky and Ortiz [23] 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 [23]), 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 [24] 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. He emphasizes that the BCC connectivity is never used and instead applies Delaunay tessellation. That is, only the adaptive BCC lattice is used to obtain an initial guess for their vertex positions.

Shimada and Gossard [25] packed spheres (or ellipsoids for anisotropic mesh generation [26, 27]) 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. [28] 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 [29].

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 [30] or GDMs [31] which have been used to triangulate isosurfaces, see e.g. [32]. 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 [33]. 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 [34]. Kobbelt et al. [35] 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. To improve robustness, Wood et al. [36] replaced the spring forces with a modified Laplacian smoothing restricted to the tangential direction. Ohtake and Belyaev [37] advocated moving the triangle centroids to the zero isocontour instead of the nodes, and matching the triangle normals with the implicit surface normals.

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 [38], 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. [39] *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.

In two spatial dimensions, Gloth and Vilsmeier [40] also moved the mesh in a direction that opposed the element quality. 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.

Human modeling is a popular application of meshing algorithms. Researchers in surgical simulation, medical device design and biomechanics of movement all require detailed models of the human form. Given the complexity of biological materials, these models are most often obtained from actual subjects via non-invasive scanning technologies or instead by dissection [41–44]. However, this data is usually voxelized or pixelized and must be converted into a more suitable format. In [44] and [41] MRI images were manually labeled to create volumetric muscle meshes for accurate muscle length and moment arm computations for gait correction surgery diagnosis. Ref. [43] used the segmented Visible Human data set to create volumetric muscle from which they designed simplified muscle models for computing accurate muscle paths in the upper extremity.

## 3 The BCC lattice

The first step in our meshing algorithm is to cover a suitable bounding box of the object with a uniform lattice. To select the particular lattice, we turn our attention to the physical world and use a body-centered cubic (BCC) tetrahedral lattice. This mesh has numerous desirable properties and is an actual crystal structure ubiquitous in nature. It appears in substances with vastly different material properties such as lithium, a soft malleable metal, and iron which is much harder and more rigid, see e.g. [11]. Other spatial tilings are possible. Üngör [45] 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 nodes located at the cell centers. These node locations may be viewed as belonging to two interlaced grids. The set of edges in the BCC lattice comprises the edges of both of these interlaced grids. Additional edge connections are also

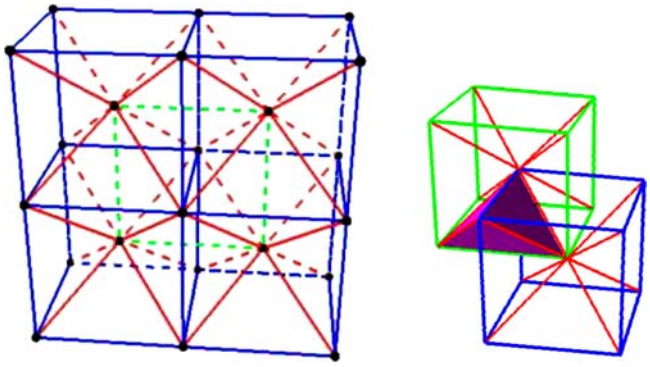

**Fig. 1** a portion of the BCC lattice. The blue and the green connections depict the two interlaced grids, and the eight red connections at each node lace these two grids together

made between a node and its eight nearest neighbors in the other grid. See Fig. 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 composed entirely of subdivision invariant tetrahedra. Meshes based on this lattice are therefore easily refined either initially or during the calculation. Each regular BCC tetrahedron can be refined into eight tetrahedra, shown in red in Fig. 2, with a one to eight (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. Fuchs demonstrates that the BCC tetrahedron is the subdivision invariant tetrahedron which differs from an equilateral one as little as possible [24].

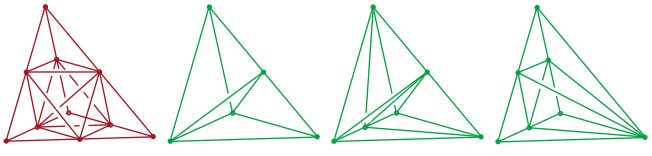

**Fig. 2** The standard red refinement (*left*) produces eight children that reside on a BCC lattice that is one half the size. We allow three types of green refinement (depicted in green)

## 4 A red green hierarchy

For many applications, computation time and memory restrictions necessitate adaptivity. In particular, volumetric simulations do not require and cannot afford a uniformly high resolution mesh. Many material response 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 a Cartesian grid, a 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. [46, 47, 23] (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. [48–50]. The general idea of a red green hierarchy is to regularly (red) refine any tetrahedron where more resolution is required, and then irregularly (green) refine tetrahedra to restore the mesh to a valid simplicial complex. 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 tetrahedra are not refined to the same level. One strategy to deal with these T-junctions is to redistribute the forces on them to the active computational nodes and to enslave their motion [51]. However, we choose the more traditional approach that eliminates them. The red tetrahedra with T-junctions are irregularly refined into fewer than eight children by introducing special case children (fewer than eight). These children are labelled green, and are of lower quality than the red tetrahedra that are part of the BCC mesh. A green tetrahedron is never refined. 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 and in the case of zero nothing needs to be done). We only allow the green refinements shown in Fig. 2. We add extra edge midpoints if necessary to arrive at one of these allowable configurations. Specifically, we only allow green tetrahedra with one, two, or three edges bisected. In the case of two bisected edges we require that these two edges are not incident to a common vertex. If they are, we add a

third midpoint. Also in the case of three bisected edges we require the three bisected edges to be on a common face of the tetrahedron. These restrictions (where all triangles are either bisected or quadrisected) smooth the gradation further and guarantee higher quality green tetrahedra. While there can 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 of refinement during simulation based on *a posteriori* error estimates. 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 Sect. 6), since part of the red parent may be missing. To further refine this tetrahedron, the green family must 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 Sects. 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

Medical data such as the National Library of Medicine's Visible Human data set often comes in volumetric form [1]. 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 [52, 53]. 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 [54]. In this section, we briefly review some basic properties of level sets and then describe how they are used as inputs to our mesh generation algorithm.

A level set defines the object implicitly with a scalar function $\varphi(x)$ defined over all of space. Its interior is the set $\{x{:}\varphi(x) < 0\}$. Similarly, its exterior is the set $\{x{:}\varphi(x) > 0\}$ and its boundary is $\{x{:}\varphi(x) = 0\}$. We further require that $\varphi(x)$ be a signed distance function, i.e., $||\nabla\varphi|| = 1$. A signed distance function provides access to several useful geometric primitives. For example, the distance of a point to the closest point on the object boundary is $|\varphi|$, and a unit normal, $N$, is simply

$$N = \frac{\nabla\phi}{\|\nabla\phi\|} = \nabla\phi,$$

where the second of the two equalities comes from $||\nabla\varphi|| = 1$.

In general, $\varphi$ is not defined analytically, but is discretized on either a uniform Cartesian grid [55] or an octree [56, 57]. In the octree case, we constrain values of fine grid nodes at gradation boundaries to match the coarse grid interpolated values, see e.g. [58]. Note that this is related to the node enslavement ideas of [51]. If a cell is adjacent to a coarser level cell, the corresponding $\varphi$ values are reset to be the value that is interpolated from the values at the coarser level of the octree. For example, if an edge of a coarser cell has values $\varphi_a$ and $\varphi_b$ at its endpoints, and this edge has a node at its midpoint because it also abuts a finer level of the octree, then $\varphi$ at the midpoint is constrained to the value $\varphi_{\text{midpoint}} = (\varphi_a + \varphi_b)/2$, which ensures a continuous level set function between different gradation levels of the octree.

When generating a tetrahedron mesh, if the signed distance function has a resolution much higher than that of our desired mesh, we apply motion by mean curvature to smooth the high frequency features. The mean curvature of a level set is defined as

$$\kappa = \nabla \cdot N = \frac{\nabla\phi}{\|\nabla\phi\|} = \nabla\phi,$$

where, again, the final equality follows from $||\nabla\varphi|| = 1$. Motion by mean curvature is implemented with a velocity of $V = -\,b\,\kappa\,N$ $(b > 0)$, i.e. the level set evolution equation is

$$\phi_t = b\kappa\|\nabla\phi\|.$$

After applying motion by mean curvature, $\varphi$ is no longer a signed distance function, so we reinitialize $\varphi$ by iterating the equation

$$\phi_\tau + S(\phi_o)(\|\nabla\phi\| - 1) = 0,$$

in fictitious time, $\tau$, until a steady state is reached [55]. Here $\varphi_o$ represents the original values of $\varphi$ at $\tau = 0$ and $S(x)$ is a smoothed sign function. For our level set preprocessing, we interleave a reinitialization procedure with every few steps of motion by mean curvature.

For our mesh generation procedure, the level set is used to guide both the refinement phase and the compression phase of the algorithm. To obtain a finer mesh near the boundary, one simply refines tetrahedra that include portions of the interface where $\varphi = 0$. If a tetrahedron has nodes with positive values of $\varphi$ and nodes with negative values of $\varphi$, it obviously contains the interface and can be refined. Otherwise, the tetrahedron is guaranteed not to intersect the interface if the minimum value of $|\varphi|$ at a node is larger than its longest edge length (tighter estimates are available, of course). The remaining cases are checked by sampling $\varphi$ appropriately (at the level set grid size $\Delta x$), allowing refinement if any sample is close enough to the interface ($|\varphi| < \Delta x$). Figure 3 shows a sphere adaptively refined near its boundary. Note how the interior mesh can still be rather coarse.

One may also wish to adaptively refine in regions of high curvature. Note that the mean curvature,

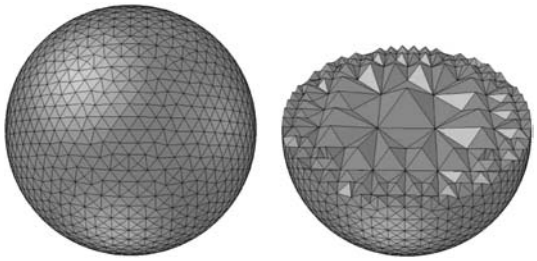

**Fig. 3** Tetrahedral mesh of a sphere (18 K elements). The cutaway view illustrates that the interior mesh can be fairly coarse even if high resolution is desired on the exterior boundary

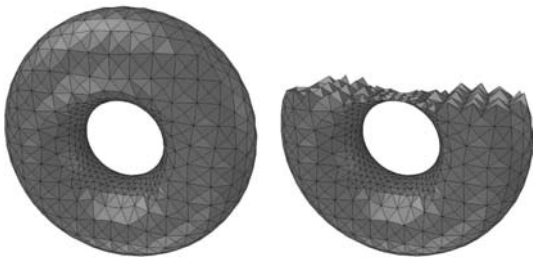

**Fig. 4** Tetrahedral mesh of a torus (8.5 K elements). Using the principal curvatures increases the level of resolution in the inner ring

$\kappa = (k_1 + k_2)/2$, is the average of the principal curvatures, $k_1$ and $k_2$. Although this is the simplest curvature measure to compute, it is an insufficient measure since it 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,

$$H = \begin{pmatrix} \phi_{xx} & \phi_{xy} & \phi_{xz} \\ \phi_{xy} & \phi_{yy} & \phi_{yz} \\ \phi_{xz} & \phi_{yz} & \phi_{zz} \end{pmatrix},$$

and projecting out the components in the normal direction via the projection matrix $P = I - NN^T$. Then the eigenvalues of $PHP/\,||\nabla\,\varphi\,||$ 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. [59]. 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 Selecting a candidate mesh

This section describes how we use the previously discussed BCC lattice, red green hierarchy, and input level set to generate a candidate mesh for the object that is ready for the final compression phase of the algorithm. Here the final topology of the mesh is selected. The final stages of the algorithm only adjust the positions of the nodes, not the connectivity.

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 $\varphi$ values all larger than the maximum edge length are removed. Any such tetrahedron is guaranteed to lie completely outside of the object.

In the next step, the remaining tetrahedra are refined according to any user defined criteria, such as indicator variables or geometric properties. We limit refinement to a user-specified number of levels. We have experimented with using both the magnitude of $\varphi$ and various measures of curvature as discussed in the previous section. Using simply the magnitude of $\varphi$ 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 allows 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,

$$|k_1|$$

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.

If the triangle mesh that is the boundary of the tetrahedra is non-manifold, it is impossible for it to match the boundary of the object which is necessarily manifold. The remaining two conditions ensure that the mesh is robust to subsequent deformations. Boundary forces can readily crush tetrahedra with all nodes on the boundary. Flattening or indenting the portion of the surface where there is a tetrahedron with all four nodes on the boundary is impossible without actually crushing or inverting it. Similarly, an interior edge (i.e. one that is not a segment of the boundary mesh) with both endpoints on the boundary makes it impossible to indent the surface between those two nodes.

Our strategy to satisfy these conditions is to select all the tetrahedra incident on a set of "enveloped" nodes that are sufficiently interior to the zero isocontour. This guarantees that every tetrahedron is incident on at least one interior node (thus automatically satisfying the

second condition above). Specifically, we envelop the set of nodes where $\varphi < 0$ that have all their incident edges at least 25% inside the zero isocontour as determined by linear interpolation of $\varphi$ along the edge. This also tends to avoid the bad interior segments and encourage a manifold boundary for reasonably convex regions, i.e. regions where the geometry is adequately resolved by the nodal samples.

Additional processing is used to guarantee appropriate topology even in regions where the mesh may be under-resolved. In a first pass, we bisect any remaining interior edge and all edges incident on non-manifold nodes. The red green procedure is then used to remove all T-junctions. If any refinement is necessary, we recalculate the set of enveloped nodes and their incident tetrahedra as above. In subsequent passes, any non-manifold node and the deeper (i.e. smaller $\varphi$) end of any interior segment are added to the enveloped set. 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 check that these additions do not create more problems, continuing to add boundary nodes to the set of enveloped nodes until we have met all requirements. Typically at most two passes are necessary. This quickly and effectively results in a candidate mesh of high quality elements 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. See Fig. 5 for an example candidate mesh.

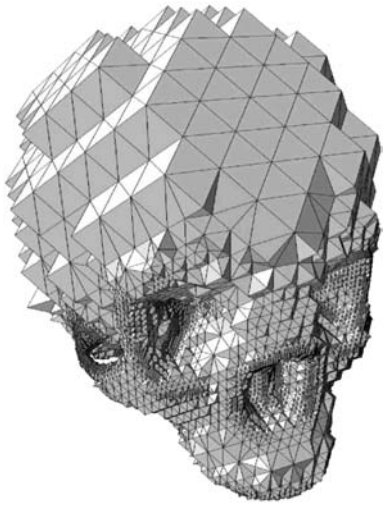

**Fig. 5** Candidate mesh of the cranium

## 7 Physics based compression

Now that we have a candidate mesh with nicely shaped elements that closely matches the boundary of the object, it remains to compress the boundary of the tetrahedron mesh to the zero isocontour of the level set. If the mesh boundary were simply snapped to the boundary of the signed distance function, the element quality would be severely degraded. To ameliorate this, we adjust the positions of the interior nodes to compensate for the motion of the boundary and to maintain mesh quality. Here we simulate the object as an elastic body with boundary conditions that drive the mesh boundary to conform to the level set boundary.

To this end, we outfit our candidate mesh with a deformable model based on either masses and springs or the finite element method. The two techniques differ in how the external forces are computed, but both have equilibrium positions that try to maintain high quality tetrahedra.

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. Specifically, the magnitude is scaled by $\varphi$, so the further the node is from the zero isocontour, the harder it is driven towards it. Also, the direction switches as the zero set is crossed because $\varphi$ is negative inside and positive outside. Using the mesh normal rather than the implicit surface normal is important because it actually tends to smooth out jagged parts of the candidate mesh, whereas $\nabla\varphi$ may cause the elements to fold over themselves.

Both physics based compression techniques calculate a set of forces that are applied to the nodes. Then, Newton's second law $F = ma$, must be integrated forward in time. This is done using a slightly-modified version of the central Newmark scheme (see e.g., [60, 61]). The elastic forces (those that do not depend on velocity) are treated explicitly, whereas the damping (velocity-dependent) forces are treated implicitly. Let $x$ represent the node positions, $v$ their velocities, and $a$ their accelerations. Then, the time-stepping proceeds as follows:

- $v^{n+1/2} = v^n + ((\Delta t)/2)a(t^n, x^n, v^{n+1/2})$ (implicit update)
- Modify $v^{n+1/2}$ in place to limit strain, strain rate, etc. as described below
- $x^{n+1} = x^n + \Delta t v^{n+1/2}$ (explicit update)
- $v^{n+1} = v^n + ((\Delta t)/2)(a(t^n, x^n, v^n) + a(t^{n+1}, x^{n+1}, v^{n+1}))$ (implicit update)

– Modify $v^{n+1}$ in place to limit strain, strain rate, etc. as described below.

This time-stepping scheme is basically a second order accurate leap frog scheme on position combined with a second order accurate trapezoidal rule for velocity. It is stable for $\Delta t < O\left(\Delta x\sqrt{\rho/k^e}\right)$, where ρ is the material density and $k^e$ is the material stiffness. In particular, it circumvents stringent quadratic, i.e. $O(\Delta x^2)$, 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.

The reason for the first implicit time step on velocity is that using an explicitly-half-stepped velocity for the position update may not be monotone, even though it is stable. This may introduce spurious oscillations in calculations which are mollified by using the implicit velocity at time $t^{n+1/2}$ for the position update instead. For more details see [61].

In the second and fifth steps of the integration algorithm, 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 [62]. 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.

The time step restriction imposed for stability is less restrictive for meshing as material parameters can be taken to be considerably softer than during most practical deformable object simulations.

### 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 [63, 64]. Löhner et al. solved the compressible Euler equations using variable spring stiffnesses to distribute the error evenly over the solution domain [65]. Later, [66] 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 [67–69]. Bossen and Heckbert point out that inter-nodal forces which both attract and repel (like springs with nonzero rest lengths) are superior to Laplacian smoothing where the nodes only attract each other [70]. 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. [71] proposed a pseudo-pressure term, [72] used an elastic (only, i.e. no damping) force emanating from the barycenter of the tetrahedron. [73] 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. Suppose that for a given tetrahedron, $x_i$ and $v_i$ for $i=1,2,3,4$, represent the positions and velocities of the four nodes. Consider the altitude from the fourth node as the apex of the tetrahedron. Let $\hat{n}$ be a unit normal to the face determined by $x_1,x_2,x_3$. Also, let $w_1,w_2,w_3$ be the barycentric coordinates of $x_{\text{int}}$, the projection of $x_4$onto the plane determined by $x_1,x_2,x_3$. In particular,

$$x_{\text{int}} = (I - \hat{n}\hat{n}^T)x_4 = w_1x_1 + w_2x_2 + w_3x_3.$$

Similarly, we have $v_{\text{int}} = w_1v_1 + w_2v_2 + w_3v_3$. Then, the elastic force due to this altitude spring is

$$f^{(e)} = \frac{k^e}{l_o}(||x_4 - x_{\text{int}}|| - l_o)\hat{n}$$

and the damping force is

$$f^{(d)} = \frac{k^d}{l_o}\hat{n}\hat{n}^T(v_4 - v_{\text{int}}),$$

where $l_o$ is the restlength of the altitude, $k^e$ is the material stiffness and $k^d$ is the material damping parameter.

These forces must then be distributed among the tetrahedron nodes. This is done according to the barycentric weights of the fictitious node. The elastic forces are $f_i^{(e)} = w_i f^{(e)}$ for $i=1,2,3$ and $f_4^{(e)} = -f^{(e)}$, and the damping forces are similarly redistributed $f_i^{(d)} = w_i f^{(d)}$ for $i=1,2,3$ and $f_4^{(d)} = -f^{(d)}$. Note that an analysis similar to that carried out in [61] reveals that, up to scaling, any other forces used (i.e. apart from the normal direction with barycentric weights) must either violate conservation of linear and angular momentum, or must act to deform the base triangle.

This model has damping forces that are linear in the nodal velocities,

$$\begin{pmatrix} f_1^{(d)} \\ f_2^{(d)} \\ f_3^{(d)} \\ f_4^{(d)} \end{pmatrix} = \frac{-k^d}{l_o} \begin{pmatrix} w_1\hat{n} \\ w_2\hat{n} \\ w_3\hat{n} \\ -\hat{n} \end{pmatrix} \begin{pmatrix} w_1\hat{n} \\ w_2\hat{n} \\ w_3\hat{n} \\ -\hat{n} \end{pmatrix}^T \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix}.$$

Moreover, because the damping matrix is a scaled outer product matrix, the damping forces are symmetric

and negative semi-definite in the nodal velocities. This allows the damping terms to be integrated using a fast conjugate gradient (CG) solver for implicit integration. For additional efficiency, we compute and cache the position-dependent components of the damping forces that are constant throughout the CG iterations. Optional accelerations for altitude springs include the ability to use only the shortest altitude spring in a given tetrahedron or the ability to only use altitude springs that are compressed beyond a threshold of their rest lengths. Note that these acceleration structures may also be computed and cached before the CG iterations.

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 $\sqrt{k^e/ml_o}$ (note our "spring constant" is $k^e/l_o$), so the sound speed scales as $l_o\sqrt{k^e/ml_o} = \sqrt{k^e l_o/m}$. Requiring the sound speed to be a material property implies that $k^e$ 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

Another physics based technique for this compression phase is to discretize the equations of continuum mechanics with the finite element method. The equations of elasticity are a more natural and more flexible way of encoding a materials response to distortion. In discretized finite element form, they resist the three-dimensional distortion of elements. A big advantage of finite element techniques over mass spring networks is the versatility provided by the framework. Finite elements allow for an arbitrary constitutive model. In this section we describe the use of finite elements during this compression phase of our algorithm as well as discuss the constitutive model implied by the altitude springs described in the previous section.

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.

The simplest example of a hyperelastic material is the St. Venant-Kirchhoff model, or linear elasticity. For this, we use the nonlinear Green strain tensor to measure the deformation of the object. Let $F = \partial x \,/\, \partial u$ be the deformation gradient where $x(u)$ represents a point's position in world coordinates as a function of its coordinates in material coordinates. Then, the Green Strain is defined via $G = 1/2(F^TF - I)$. Isotropic, linearly-elastic materials have a stress strain relationship of the form

$$S_e = \lambda \mathrm{tr}(G)I + 2\mu G$$

where λ and μ are the Lamé coefficients. Damping stress is modeled similarly with $S_d = \alpha \mathrm{tr}(\nu)\ I + 2\beta\ \nu$, where $\nu = \partial G/\partial t$ is the strain rate. The total second Piola-Kirchhoff stress tensor is then $S = S_e + S_d$.

Although simple, St. Venant-Kirchhoff elasticity is not very practical outside of the small strain regime [74]. We have found that a more robust constitutive model for large deformation simulations is the one for neo-Hookean materials. For this constitutive model, it is more natural to use the left Cauchy-Green tensor $b = FF^T$ to measure the deformation. For neo-Hookean materials, the Cauchy stress is

$$\sigma = \frac{\mu}{J}(b - I) + \frac{\lambda \ln J}{J} I,$$

where $J = \det F$ is the relative volume change (i.e. it is the ratio between the volume of the deformed tetrahedron to the volume of the undeformed tetrahedron). This constitutive model increases the resistance to deformation as an element undergoes an excessive change in volume.

The neo-Hookean hydrostatic pressure term ($(\lambda\ lnJ/J)I$) little effect on deformations that distort an element's quality (i.e aspect ratio, min and max dihedral angles) while causing minimal change in volume. Altitude springs work more efficiently to preserve an element's quality. The finite volume formulation of [75] allows us to interpret the forces created by altitude springs as an equivalent internal stress of the form

$$\sigma = \sum_{i=1}^{4} \frac{3f_i}{A_i} n_i n_i^T,$$

where $A_i$ are the areas of the faces of a given element, $n_i$ are unit vectors in the directions of the tetrahedron altitudes and $f_i$ are the magnitudes of the altitude spring forces (see appendix −1). This is a more attractive response to element deformation than the neo-Hookean hydrostatic pressure because the stress works to retain mesh quality by preserving *shape* rather than volume.

To discretize these constitutive models, we use finite elements with linear basis functions in each tetrahedron. The displacement of material is a linear function of the tetrahedron's four nodes. >From the nodal locations and velocities we obtain the Jacobian of this linear mapping, $F$, and its derivative, $\dot{F}$,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. See [75] for the specific finite volume formulation that we use.

## 8 Optimization based compression

As an alternative to—or as an additional step before or after—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 Sect. 6), this produces excellent results. Freitag and Ollivier-Gooch [76] 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 tends to gets stuck in local extrema. This could be overcome with more global optimization techniques than our simple node-by-node greedy sweep, but it appears to be simpler, more robust, and faster to simply use several (roughly ten) greedy sweeps to match the boundary. 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 (this ordering is determined by a simple breadth-first search from the boundary nodes). 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 that begin by moving the boundary nodes 1/3 of the signed distance in the mesh normal direction, then finish off with five to ten sweeps where boundary nodes are moved 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 (Figure 6, 7).

While more efficient gradient methods may be used for the nodal optimization, we found a simple pattern search (see e.g. [77]) to be attractive for its robustness, simplicity of implementation, and flexibility in easily accommodating any quality metric including non-smooth max/min metrics. 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, thus not allowing the optimization to consider nodal positions outside of the tangent plane. 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 $\theta_M$ is the maximum angle between face normals. For surface nodes we add to this an additional 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 $\psi_M$ 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. We have not performed a detailed study of which metrics perform better, and thus expect the optimization results could be improved in general and especially for particular applications where a specific metric is important (e.g. geometry factors arising from finite element error analysis).

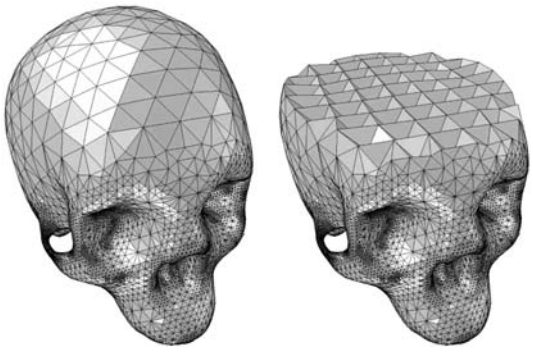

**Fig. 6** Tetrahedral mesh (*left*) and cutaway view (*right*) of a cranium (80K elements)

## 9 Discussion

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 (ranging from a few minutes to a few hours on the largest meshes) than the mass spring methods, but produce 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

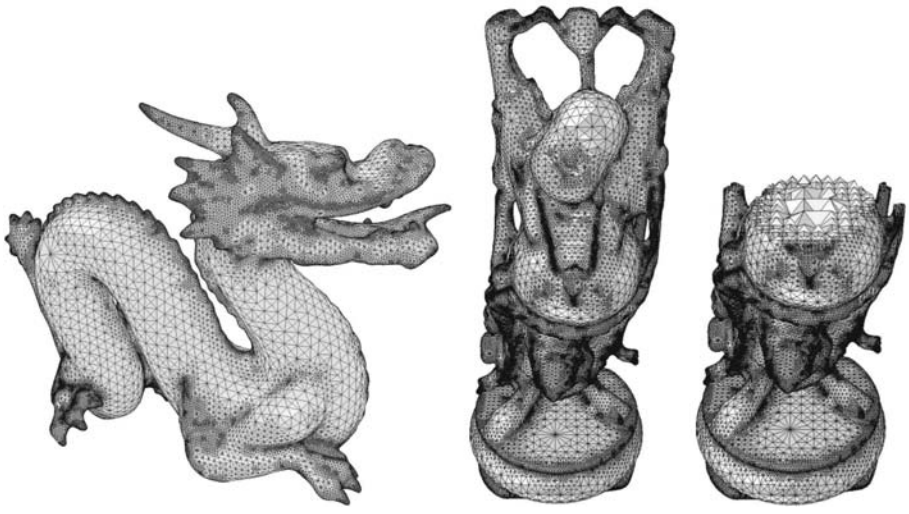

**Fig. 7** Tetrahedral mesh and cutaway view of a model Dragon (500 K elements) and Buddha (800 K elements)

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 average 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.

## 10 Example: muscle simulation

Musculoskeletal simulation is an active research area in biomechanics of movement, biomedical device design, surgery simulation and computer graphics. 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 [1]. The data for these biological materials are originally voxelized in the form of a

**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

| 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 |

segmented series of consecutive images that can be used to create a level set description of each tissue geometry. The level set representation of tissue and bone geometry is particularly useful for correcting many problems inherent in the data. The voxelized data is considerably aliased, but this can be repaired with a few time steps of motion by mean curvature in most cases. Also, level sets are very well suited for constructive solid geometry operations. Much of the segmented data that we have is incomplete due to the difficulty of the segmentation process. However, using basic geometric primitives we have been able to rebuild parts of tissues with CSG using anatomical texts as a reference [8]. Once a satisfactory implicit representation has been obtained, it can then be used with either the dynamic or optimization based algorithm. Figure 9 shows adaptive resolution muscles, tendons and bones in the upper extremity that were created using dynamic meshing with a finite element constitutive model. The bones are modeled as rigid bodies and were created using the extension of our algorithm to surfaces (see Sect. 11).

Before any muscle simulation can be performed, a kinematic structure for the skeleton must be developed to set boundary conditions at the tendonous muscle attachments. The joints in the shoulder girdle are particularly intricate and involve a complex coupling of degrees of freedom through the glenohumeral and acromioclavicular joints. Our meshing algorithm allows us to create models of the bones that resolve sub-millimeter anatomical detail. This resolution allows us to make use of landmarks on the bone geometries that can be used to set up local coordinate frames as was done in [42].

We simulate both 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 fasicles embedded in a matrix of isotropic material [78], and we use a nonlinear transversely-isotropic quasi-incompressible constitutive model [75, 79, 80] 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 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 [81]. 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 [82, 83] 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 [84] to compute activations of redundant sets of muscles. These computations are performed using a simplified muscle model (see Fig. 10) that treats muscle as a piecewise linear band that wraps around various geometric primitives representing muscle/muscle and muscle/bone collision.

Fig. 8 shows sample frames from our musculo-skeletal simulations. The images on the left depict 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. The rightmost images in Fig. 8 show several frames of musculo-skeletal motion. The motion of the kinematic skeleton was key-framed (although our framework allows for motion data from other sources such as 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 12 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 Surface meshing

The general theme of our algorithm is applicable in any dimension and on general manifolds. First, we tile ambient space as regularly as possible, then we select a subset of elements that are nicely connected and roughly conform to the object, and finally we deform them to match the boundary. Figure 13 shows the result of triangulating a two-dimensional manifold with boundary. We began with a surface mesh of the dragon

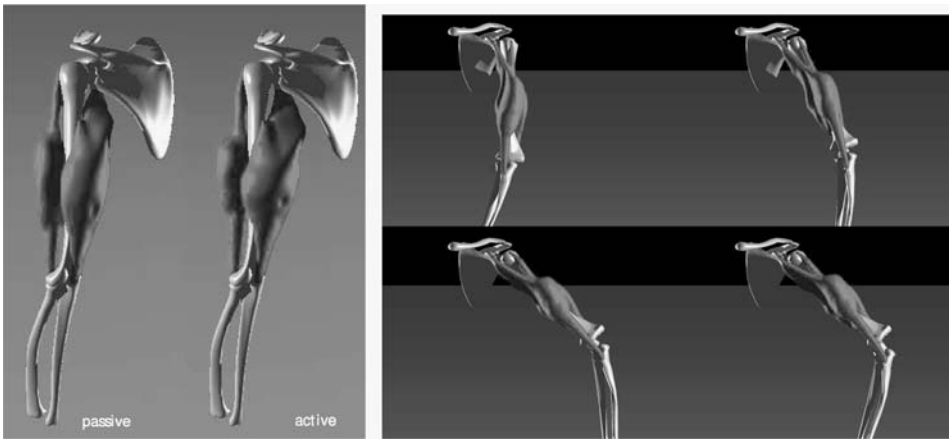


**Fig. 8** The figure on the left depicts a posterior (*from behind*) view of the upper arm and shows contraction of the triceps muscle and the partially occluded biceps muscle from passive (*left*) to full activation (*right*). The figure on the right demonstrates muscle contraction with underlying skeletal motion

**Fig. 9** Adaptive resolution meshes of muscles in the arm

actually created as the boundary of a tetrahedral mesh from our method with additional subdivision, edge-swapping, and nodal smoothing. See [40] for example. We found that this simple technique avoids topological difficulties that may be encountered in simpler level set surface triangulation algorithms. We further interactively specified a ''trimming'' level set indicating areas of the surface to trim away, using a sculpting tool described in [85]. We discarded a subset of the surface triangles inside the trimming level set using the same topological considerations that are used in selecting candidate volumetric tetrahedral meshes. This candidate surface mesh was then driven to the trimming boundary using physics based compression with masses and springs. At every time step of the compression, the mesh nodes were projected back onto the surface of the dragon to ensure a good match to the geometry. This technique is useful because high quality two-dimensional surface meshes are important for well-conditioned thin shell simulations of objects such as fascias, membranes, skin, and cloth.

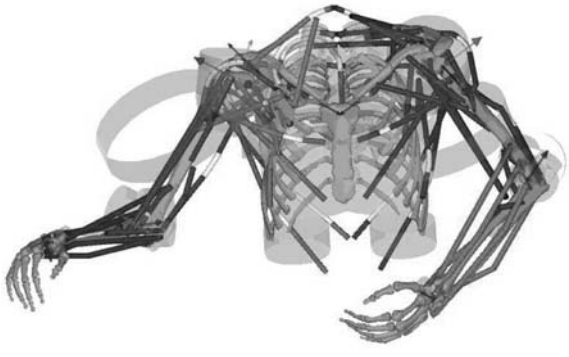

**Fig. 10** Inverse kinematics muscles

## 12 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 tetrahedral mesh designed specifically for large deformation. Key points of our algorithm that make it particularly well suited for large 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

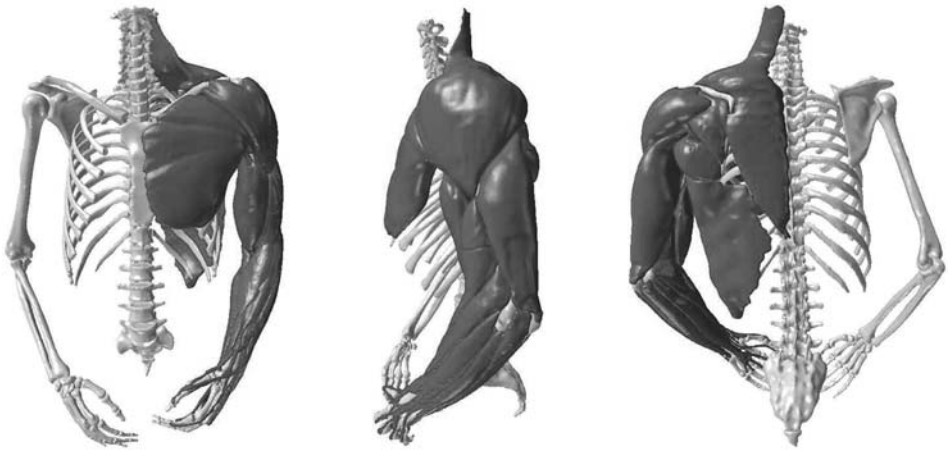

**Fig. 11** Muscles of the upper limb

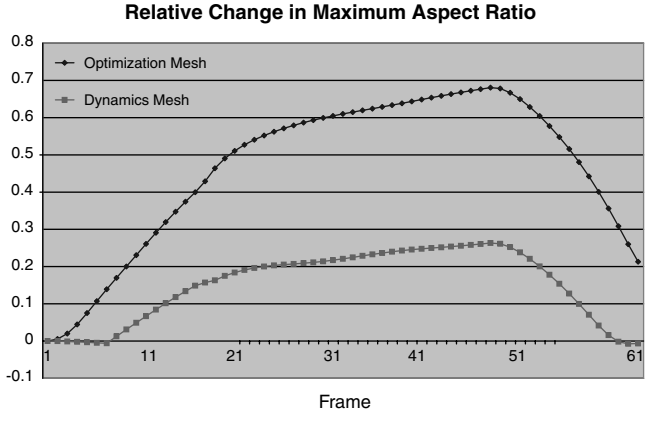


**Fig. 12** Plot of changes in maximum aspect ratios during simulation of isometric contraction for dynamics and optimization based meshes

(in fact this is what originally led us to the problematic connectivity).

Also, we developed a way to augment the use of masses and springs in mesh generation with altitude springs to prevent elements from collapsing. This formulation results in damping forces which are linear in the nodal velocities. Moreover, the damping matrix is symmetric negative semi-definite, which allows an efficient conjugate gradient solver to be used for implicit time integration.

Finally, we demonstrated the strength or our algorithm by simulating deformable volumetric skeletal muscle and its interaction with an underlying skeleton. We created level set representations of soft tissues and bones from the segmented Visible Human data set. The deformable tissues were constructed using the tetrahedron meshing algorithms and the rigid bones were done using the extension of the algorithm to surfaces. The initial implicit representation of the anatomical structures allowed us to make use of standard level set procedures to smooth out inherent aliasing as well as to perform basic constructive solid geometry procedures to repair errors in the segmented data.

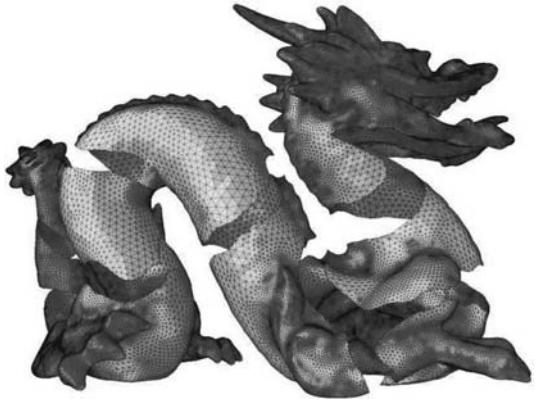

**Fig. 13** Our method is easily extended to create a triangle mesh of a manifold with boundary. The peeled away regions of the dragon were modeled with a second level set

**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, ARO DAAD19-03-1-0331, NSF ITR-0121288, NSF ACI-0205671, NSF DMS-0106694, NSF ACI-0323866 and NSF IIS-0326388. 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.

## Appendix

Consider the tetrahedron defined by the points $x_1,x_2,x_3,x_4$ and the altitude spring from $x_4$ to the triangle $x_1,x_2,x_3$. The Cauchy stress in the tetrahedron due to this spring is

$$\sigma_{alt} = \frac{3f}{A_4} n_4 n_4^T,$$

where $f$ is the scalar force in an altitude spring (see Sect. 6), $A_4$ is the area of the triangle $x_1,x_2,x_3$ and $n_4$ is the

outward pointing normal of the tetrahedron on face $x_1,x_2,x_3$.

This can be shown by considering the results of [75]. A finite volume interpretation of the constant strain tetrahedron finite element forces shows that the nodal force response to an internal Cauchy stress $\sigma$ can be written as

$$f_i = \sigma\left(\frac{A_j n_j + A_k n_k + A_l n_l}{3}\right).$$

Using this result, the force on $x_4$ due to $\sigma_{alt}$ is

$$f_4 = fn_4\left(\frac{A_1 n_4^T n_1}{A_4} + \frac{A_2 n_4^T n_2}{A_4} + \frac{A_3 n_4^T n_3}{A_4}\right).$$

A simple geometric argument shows that $-(A_1 n_4^T n_1)/A_4$ is the ratio of the projected area of the face $x_2,x_3,x_4$ on the plane through $x_1,x_2,x_3$ with $A_4$. This is the barycentric weight of $x_4$ in the plane $x_1,x_2,x_3$ on the point $x_1$. Defining the barycentric weight as $w_1$, we can similarly derive

$$\frac{A_2 n_4^T n_2}{A_4} = -w_2,\ \frac{A_3 n_4^T n_3}{A_4} = -w_3.$$

Therefore, the force on $x_4$ is

$$f_4 = fn_4(-w_1 - w_2 - w_3) = -fn_4.$$

Similarly,

$$\begin{aligned}
f_1 &= fn_4\left(\frac{A_2 n_4^T n_2}{A_4} + \frac{A_3 n_4^T n_3}{A_4} + \frac{A_4 n_4^T n_4}{A_4}\right) \\
&= fn_4(1 - w_2 - w_3) = fn_4 w_1 \\
f_2 &= fn_4\left(\frac{A_1 n_4^T n_1}{A_4} + \frac{A_3 n_4^T n_3}{A_4} + \frac{A_4 n_4^T n_4}{A_4}\right) \\
&= fn_4(1 - w_1 - w_3) = fn_4 w_2 \\
f_3 &= fn_4\left(\frac{A_1 n_4^T n_1}{A_4} + \frac{A_2 n_4^T n_2}{A_4} + \frac{A_4 n_4^T n_4}{A_4}\right) \\
&= fn_4(1 - w_1 - w_2) = fn_4 w_3,
\end{aligned}$$

which are precisely the altitude spring forces.

## References


1. U.S. National Library of Medicine (1994) The Visible Human Project. http://www.nlm.nih.gov/research/visible/
2. Martins J, Pires E, Salvado R, Dinis P (1998) A numerical model of passive and active behavior of skeletal muscles. Comput Meth Appl Mech Eng 151:419–433
3. Hirota G, Fisher S, State A, Lee C, Fuchs H (2001) An implicit finite element method for elastic solids in contact. Comput Anim
4. Cotin S, Delingette H, Ayache N (1996) Real-time volumetric deformable models for surgery simulation. Proc of Vis in Biomed Comput, pp 535–540
5. Ganovelli F, Cignoni P, Montani C, Scopigno R (2000) A multiresolution model for soft objects supporting interactive cuts and lacerations. Eurographics, pp 271–282
6. Bro-Nielsen M, Cotin S (1996) Real-time volumetric deformable models for surgery simulation using finite elements and condensation. Comput Graph Forum 15(3):57–66
7. Fleishmann P, Kosik R, Selberherr S (1999) Simple mesh examples to illustrate specific finite element mesh requirements. In: 8th international meshing roundtable, pp 241–246
8. Marcum DL (1995) Generation of unstructured grids for viscous flow applications. AIAA
9. Garimella R, Shephard M (1998) Boundary layer meshing for viscous flows in complex domains. In: 7th international meshing roundtable, pp 107–118
10. Lohner R, Cebral J (1999) Generation of non-isotropic unstructured grids via directional enrichment. In: 2nd symposium on trends in unstructured mesh generation
11. Burns G, Glazer AM (1990) Space groups for solid state scientists, 2nd edn. Academic, New York
12. Osher S, Sethian J (1988) fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. J Comp Phys 79:12–49
13. Weatherill NP, Hassan O (1994) Efficient three-dimensional delaunay triangulation with automatic point creation and imposed boundary constraints. Int J Numer Meth Eng 37:2005–2039
14. Shewchuk J (1998) Tetrahedral mesh generation by delaunay refinement. In: Procedings of the 14th annual symposium on computer and geomics, pp 86–95
15. Cheng SW, Dey TK, Edelsbrunner H, Facello MA, Teng SH (2000) Sliver exudation. J ACM 47(5):883–904
16. Edelsbrunner H, Guoy D (2002) An experimental study of sliver exudation. Eng Comput 18(3):229–240
17. Shewchuk J (2002) Constrained Delaunay tetrahedralizations and provably good boundary recovery. In: 11th international meshing roundtable
18. Schöberl J (1997) NETGEN - an advancing front 2D/3D mesh generator based on abstract rules. Comput Vis Sci 1:41–52
19. Möller P, Hansbo P (1995) On advancing front mesh generation in three dimensions. Int J Numer Meth Eng 38:3551–3569
20. Lo SH (1991) Volume Discretization into tetrahedra - I, Verification and orientation of boundary surfaces. Comput Struct 39(5):493–500
21. Lo SH (1991) Volume discretization into tetrahedra - II, 3D triangulation by advancing front approach. Comput Struct 39(5):501–511
22. Mavriplis DJ (1995) An advancing front delaunay triangulation algorithm designed for robustness. J Comp Phys 117:90–101
23. Radovitzky RA, Ortiz M (2000) Tetrahedral mesh generation based in node insertion in crystal lattice arrangements and advancing-front Delaunay triangulation. Comput Meth Appl Mech Eng 187:543–569
24. Fuchs A (1998) Automatic grid generation with almost regular Delaunay tetrahedra. In: 7th international meshing roundtable, pp 133–148
25. Shimada K, Gossard D (1995) Bubble mesh: automated triangular meshing of non-manifold geometry by sphere packing. ACM 3rd symposium on solid model and application, pp 409–419
26. Yamakawa S, Shimada K (2000) High quality anisotropic tetrahedral mesh generation via packing ellipsoidal bubbles. In: 9th international meshing roundtable, pp 263–273
27. Yamakawa S, Shimada K (2003) Anisotropic tetrahedral meshing via bubble packing and advancing front. Int J Numer Meth Eng 57:1923–1942
28. Li X, Teng S, Üngör A (1999) Biting spheres in 3D. In: 8th international meshing roundtable, pp 85–95
29. Li X, Teng S, Üngör A (1999) Biting ellipses to generate anisotropic mesh. In: 8th international meshing roundtable, pp 97–108
30. Kass M, Witkin A, Terzopoulos D (1987) Snakes: active contour models. Int J Comput Vis, pp 321–331
31. Miller J, Breen D, Lorensen W, O'Bara R, Wozny M (1991) Geometrically deformed models: a method for extracting closed geometric models from volume data. Comput Graph (SIGGRAPH Proc.), pp 217–226

32. Sadarjoen IA, Post FH (1997) Deformable surface techniques for field visualization. Eurographics, pp 109–116
33. Neugebauer P, Klein K (1997) Adaptive triangulation of objects reconstructed from multiple range images. Vis
34. Grosskopf S, Neugebauer PJ (1998) Fitting geometrical deformable models to registered range images. European Workshop on 3D structure from multiple images of large-scale environments (SMILE), pp 266–274
35. Kobbelt LP, Vorsatz J, Labsik U, Seidel HP (1999) A shrink wrapping approach to remeshing polygonal surfaces. Eurographics, pp 119–130
36. Wood Z, Desbrun M, Schröder P, Breen D (2000) Semi-regular mesh extraction from volumes. Vis. pp 275–282
37. Ohtake Y, Belyaev AG (2002) Dual/primal mesh optimization for polygonized implicit surfaces. In: Proceedings of the 7th ACM Symposium on Solid Model Appl, ACM, New York, pp 171–178
38. de Figueiredo LH, Gomes J, Terzopoulos D, Velho L (1992) Physically-based methods for polygonization of implicit surfaces. In: Proceedings of the conference on graphic interface, pp 250–257
39. Velho L, Gomes J, Terzopoulos D (1997) Implicit manifolds, triangulations and dynamics. J Neural Parallel Scientif Comput 15(1–2):103–120
40. Gloth O, Vilsmeier R (2000) Level Sets as Input for Hybrid Mesh Generation. In: 9th international meshing roundtable, pp 137–146
41. Arnold A, Salinas S, Asakawa D, Delp S (2000) accuracy of muscle moment arms estimated from MRI-based musculoskeletal models of the lower extremity. Comput Aided Surg 5:108–119
42. Garner B, Pandy M (1999) A kinematic model of the upper limb based on the visible human project (VHP) image dataset. Comput Meth Biomech Biomed Eng 2:107–124
43. Garner B, Pandy M (2001) Musculoskeletal model of the upper limb based on the visible human male dataset. Comput Meth Biomech Biomed Eng 4:93–126
44. Arnold A, Blemker S, Delp S (2001) Evaluation of a deformable musculoskeletal model for estimating muscle-tendon lengths during crouch gait. Comput Aided Surg 29:263–274
45. Üngör A (2001) Tiling 3D euclidean space with acute tetrahedra. In: Proceedings of the canadian conference on Computer Geomics, pp 169–172
46. Yerry MA, Shephard MS (1984) Automatic three-dimensional mesh generation by the modified Octree technique. Int J Numer Meth Eng 20:1965–1990
47. Shephard MS, Georges MK (1991) Automatic three-dimensional mesh generation by the finite Octree technique. Int J Numer Meth Eng 32:709–739
48. Bey J (1995) Tetrahedral grid refinement. Computing 55:355–378
49. Grosso R, Lürig C, Ertl T (1997) The multilevel finite element method for adaptive mesh optimization and visualization of volume data. Visualization, pp 387–394
50. de Cougny HL, Shephard MS (1999) Parallel refinement and coarsening of tetrahedral meshes. Int J Numer Meth Eng 46:1101–1125
51. Bessette G, Becker E, Taylor L, Littlefield D (2003) Modeling of impact problems using an H-adaptive, explicit lagrangian finite element method in three dimensions. Comput Meth Appl Mech Eng 192:1649–1679
52. Tsitsiklis J (1995) Efficient algorithms for globally optimal trajectories. IEEE Trans Automat Control 40:1528–1538
53. Sethian J (1996) A fast marching level set method for monotonically advancing fronts. Proc Natl Acad Sci 93:1591–1595
54. Zhao HK, Osher S, Fedkiw R (2001) Fast surface reconstruction using the level set method. In: 1st IEEE Wrkshp on Variational and Level Set Methods 8th Int Conf on Comput Vis, pp 194–202
55. Osher S, Fedkiw R (2002) Level set methods and dynamic implicit surfaces. Springer, Berlin Heidelberg New York
56. Strain J (1999) Fast tree-based redistancing for level set computations. J Comput Phys 152:664–686
57. Strain J (1999) Tree methods for moving interfaces. J Comput Phys 151:616–648
58. Westermann R, Kobbelt L, Ertl T (1999) Real-time exploration of regular volume data by adaptive reconstruction of isosurfaces. The Vis Comput 15(2):100–111
59. Ambrosio L, Soner HM (1996) Level set approach to mean curvature flow in arbitrary codimension. J Diff Geom 43:693–737
60. Hughes T (1987) The finite element method: linear static and dynamic finite element analysis. Prentice Hall, Englewoodcliff
61. Bridson R, Marino S, Fedkiw R (2003) Simulation of clothing with folds and wrinkles. In: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Comput Anim, pp 28–36
62. Bridson R, Fedkiw R, Anderson J (2002) Robust treatment of collisions, contact and friction for cloth animation. ACM Trans Graph (SIGGRAPH Proc) 21:594–603
63. Gnoffo P (1982) A vectorized, finite-volume, adaptive-grid algorithm for Navier-stokes calculations. Num Grid Generat, pp 819–835
64. Gnoffo P (1982) A finite-volume, adaptive grid algorithm applied to planetary entry flowfields. AIAA
65. Lohner R, Morgan K, Zienkiewicz OC (1986) Adaptive grid refinement for compressible euler equations. Wiley, New York, pp 281–297
66. Nakahashi K, Deiwert GS (1987) Self-adaptive-grid method with application to airfoil flow. AIAA 25(4):513–520
67. Dompierre J, Vallet M, Fortin M, Habashi WG, Aït-Ali-Yahia D, Boivin S, Bourgault Y, Tam A (1995) Edge-based mesh adaptation for CFD. In: Conference on Numerical Methods for the Euler and Navier-Stokes Equations
68. Vallet M, Dompierre J, Bourgault Y, Fortin M, Habashi WG (1996) Coupling flow solvers and grids through an edge-based adaptive grid method. Fluids Engineering Div Conference, vol 3
69. Fortin M, Vallet M, Dompierre J, Bourgault Y, Habashi WG (1996) Anisotropic mesh adaptation: theory, validation, and applications. Comput Fluid Dynamics
70. Bossen FJ, Heckbert PS (1996) A pliant method for anisotropic mesh generation. In: Proceedings of the 5th international meshing roundtable, pp 63–76
71. Palmerio B (1994) An attraction-repulsion mesh adaption model for flow solution on unstructured grids. Comput Fluids 23(3):487–506
72. Bourguignon D, Cani MP (2000) Controlling anisotropy in mass-spring systems. Eurographics, Eurographics Assoc, pp 113–123
73. Cooper L, Maddock S (1997) Preventing collapse within mass-spring-damper models of deformable objects. In: Proceedings of the 5th international conference in central europe on computer graphics and vision
74. Bonet J, Wood R (1997) Nonlinear continuum mechanics for finite element analysis. Cambridge University Press, Cambridge
75. Teran J, Blemker S, Ng V, Fedkiw R (2003) Finite volume methods for the simulation of skeletal muscle. In: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on computer Animation, pp 68–74
76. Freitag L, Ollivier-Gooch C (1997) Tetrahedral mesh improvement using swapping and smoothing. Int J Num Meth Eng 40:3979–4002
77. Torczon V (1997) On the convergence of pattern search algorithms. SIAM J Opt 7(1):1–25
78. Fung YC (1981) Biomechanics: mechanical properties of living tissues. Springer, Berlin Heidelberg New York
79. Yucesoy CA, Koopman BH, Huijing PA, Grootenboer HJ (2002) Three-dimensional finite element modeling of skeletal muscle using a two-domain approach: linked fiber-matrix mesh model. J Biomech 35:1253–1262
80. Weiss J, Maker B, Govindjee S (1996) Finite-element impementation of incompressible, transversely isotropic hyperelasticity. Comput Meth Appl Mech Eng 135:107–128

81. Zajac F (1989) Muscle and tendon: properties, models, scaling, and application to biomechanics and motor control. Crit Rev Biomed Eng 17(4):359–411
82. Ng-Thow-Hing V, Fiume E (1997) Interactive display and animation of B-spline solids as muscle shape primitives. In: Thalmann D, van de Panne M (eds) Proceedings of the eurographics workshop on computer animation and sim. Springer, Berlin Heidelberg New York
83. Ng-Thow-Hing V, Fiume E (2002) Application-specific muscle representations. In: Sturzlinger W, McCool M (eds) Proc of Gr Inter Canadian Information Processing Society, pp 107–115
84. Crowninshield R (1978) Use of optimization techniques to predict muscle forces. Trans ASME 100:88–92
85. Bridson R (2003) Computational aspects of dynamic surfaces. PhD thesis, Stanford University