# Dynamic Local Remeshing for Elastoplastic Simulation

Martin Wicke Daniel Ritchie Bryan M. Klingner* Sebastian Burke Jonathan R. Shewchuk James F. O'Brien

University of California, Berkeley and *Graphwalking Associates

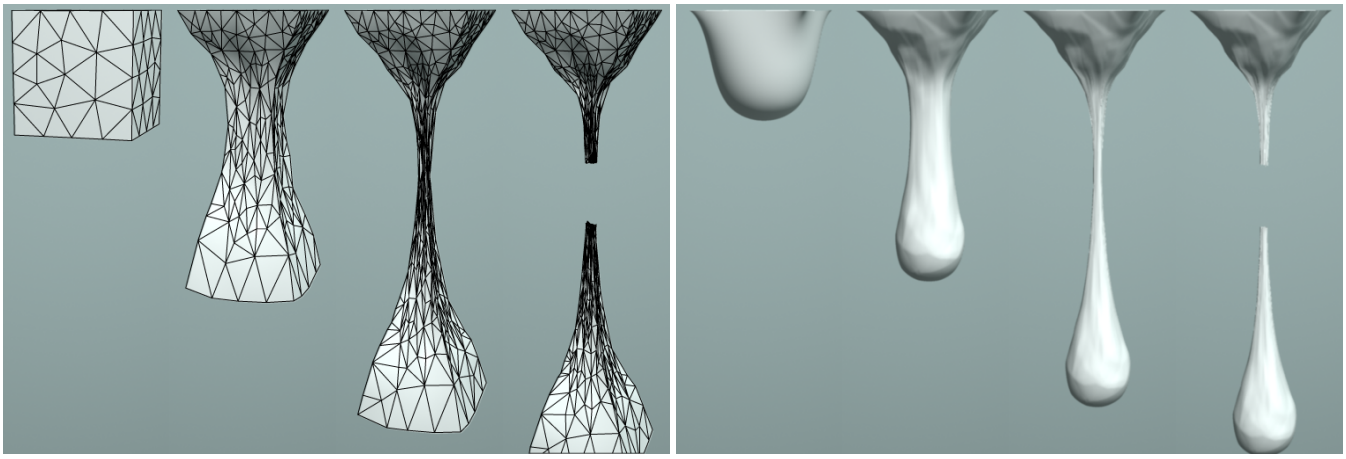


**Figure 1:** *An elastoplastic substance slowly drips from a horizontal surface. A dynamic meshing algorithm refines the drop while maintaining high-quality tetrahedra. At the narrowest part of the tendril, the mesher creates small, anisotropic tetrahedra where the strain gradient is anisotropic, so that a modest number are adequate. Work hardening causes the tendril to become brittle, whereupon it fractures. At right, we animate a fine triangulated surface embedded in the mesh.*

## Abstract

We propose a finite element simulation method that addresses the full range of material behavior, from purely elastic to highly plastic, for physical domains that are substantially reshaped by plastic flow, fracture, or large elastic deformations. To mitigate artificial plasticity, we maintain a simulation mesh in both the current state and the rest shape, and store plastic offsets only to represent the non-embeddable portion of the plastic deformation. To maintain high element quality in a tetrahedral mesh undergoing gross changes, we use a dynamic meshing algorithm that attempts to replace as few tetrahedra as possible, and thereby limits the visual artifacts and artificial diffusion that would otherwise be introduced by repeatedly remeshing the domain from scratch. Our dynamic mesher also locally refines and coarsens a mesh, and even creates anisotropic tetrahedra, wherever a simulation requests it. We illustrate these features with animations of elastic and plastic behavior, extreme deformations, and fracture.

**Keywords:** finite element simulation, dynamic meshing, local remeshing, adaptive refinement, plasticity, elastoplasticity, fracture.

**From the ACM SIGGRAPH 2010 conference proceedings.**


**ACM SIGGRAPH 2010, Los Angeles**


## 1 Introduction

Finite element simulations are increasingly used to model physical phenomena such as fracture, cutting, and plastic flow that require the finite element mesh to evolve as time progresses. Meshes are refined to capture detailed physical behavior, fractures are simulated by subdividing mesh elements, and plastic flow can be so extreme that meshes must be periodically replaced to prevent the discretization error from ballooning. See Figure 1 for an example. In car crashes, muscle movements, shattering plates, explosions, and melting candles, physical domains reshape themselves.

Traditional Lagrangian elastic simulations use a fixed material-space mesh to represent an object, and a mapping from the material mesh to world space to represent its deformation. Material strains are determined by this mapping. Extreme deformations can make the mesh elements become skinny or degenerate in world space, or even turn them inside out, in which case the simulation becomes meaningless. The addition of plastic flow to a simulation implies that the elements change shape in material space as well. Sufficient plastic flow can degrade the material space elements until their accuracy is ruined, and reshape an object so completely that a new mesh is obligatory.

A recent trend in plasticity modeling is to discard the material-space mesh. A simulation maintains a world-space mesh of the object and strain information, from which each element's rest shape can be inferred. When the elements in the world-space mesh are deformed enough to threaten the simulation's accuracy, the entire domain is remeshed from scratch [Bargteil et al. 2007; Wojtan and Turk 2008; Wojtan et al. 2009]. A disadvantage of wholesale remeshing is that it quickly accumulates large numerical errors because of the frequent need to resample physical properties such as velocity and strain from an old mesh to a new mesh. This rapid accumulation of error is called *artificial diffusion*, because the physical properties sampled on the mesh diffuse unnaturally through the material. When artificial diffusion afflicts the strain field, it manifests exag-

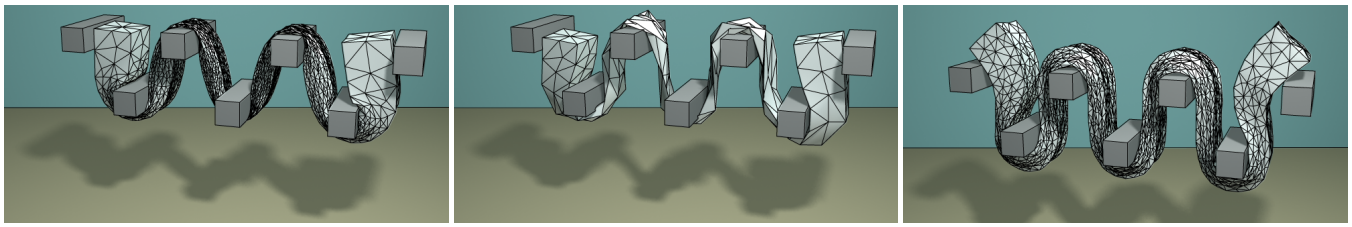


**Figure 2:** *A rectangular bar is bent by the masticator. From left to right: purely elastic material behavior with adaptive refinement; purely elastic without refinement; and with plastic flow and adaptive refinement.*

gerated plastic-like behavior even for purely elastic objects. These methods succeed for highly plastic materials in part because large plastic flows mask this error. Unfortunately, if an object undergoes an extreme deformation but only a portion of the object undergoes plastic flow, then remeshing from scratch will subtly change the rest shape of the purely elastic portion of the object, creating unsightly visual artifacts when it reverts to its new rest shape.

We propose an alternative that addresses the whole range of material behavior from purely elastic to highly plastic, by the use of *dynamic meshing*: a conservative local remeshing algorithm maintains high tetrahedron quality while limiting the accumulation of numerical error and artificial diffusion. Our simulation retains the traditional Lagrangian material-space mesh and its mapping to world space, illustrated in Figure 3. The material-space mesh is not changed by elastic deformations, but it is reshaped by plastic flow. We use local remeshing to repair degraded tetrahedra in material and not world space, so purely elastic regions of an object do not lose their original shape. Our remesher is conservative and changes as few tetrahedra as it can, so artificial diffusion is reduced everywhere.

Remeshing is triggered by mesh geometry in both the material and world spaces. We impose a minimum bound on acceptable tetrahedron quality in material space, and repair any tetrahedron that falls below the threshold as a consequence of plastic flow. Although purely elastic deformations do not deform the material space mesh, local adaptive refinement of the mesh may be necessary to accommodate large deformations in world space (so the geometry is accurately represented) or a strain field with a large gradient (so the strain is accurately interpolated), as illustrated in Figure 2. We also coarsen the mesh where it is unnecessarily fine, and use anisotropic tetrahedra where the strain field warrants them.

## 2 Background

Physically-based simulation of deformable objects was introduced to computer animation by Terzopoulos *et al.* [1987] and other contemporaneous work. A survey article by Gibson and Mirtich [1997] details much of the early work on deformable modeling, while Nealen *et al.* [2006] survey some more recent approaches.

Without remeshing, the finite element method is limited to a space of possible deformations dictated by the simulation mesh. For scenarios involving only moderate deformation, it is easy to strike a good compromise between having resolution fine enough to represent deformation accurately and coarse enough to allow fast computation times. However, large deformations involving substantial twisting, bending, and swirling require very fine meshes to represent the displacement function that maps the undeformed to the deformed configuration. This requirement is especially troublesome when one does not know beforehand how fine the mesh must be, or where the mesh must be finest.

For systems dominated by plastic flow (*e. g.* viscoelastic fluids or highly elastoplastic solids), researchers have addressed extreme deformations with Eulerian methods that do not maintain an explicit material-space reference configuration; instead, material advects through a world-space mesh. This is the approach taken by Carlson *et al.* [2002; 2004] and Goktekin *et al.* [2004]. Unfortunately, frequent resampling of the field variables during advection introduces substantial artificial diffusion that is clearly visible when modeling less plastic materials.

As we discussed in the introduction, other researchers model predominantly plastic materials with Lagrangian formulations and world-space meshes that are remeshed from scratch when necessary [Bargteil et al. 2007; Wojtan and Turk 2008; Wojtan et al. 2009]. An advantage of this approach is that it takes into account that there might not be a material-space mesh consistent with the object as a whole—that is, if all external forces were removed and the object settled to its equilibrium shape, it would still experience internal strain. (Our method retains this advantage, even though we have an explicit material-space mesh.) A disadvantage is that artificial diffusion can introduce visual artifacts in purely elastic or barely-plastic portions of an object.

Frequent remeshing from scratch is also used to model free or moving boundaries in Eulerian formulations of fluid dynamics on tetrahedral meshes [Klingner et al. 2006; Chentanez et al. 2007]. The artificial diffusion arising from resampling due to remeshing in these methods is not worse than would otherwise be incurred through resampling due to advection. In computational fluid dynamics, volume-of-fluid and moment-of-fluid methods are used to prevent diffusion due to resampling (see, *e. g.,* Kucharik *et al.* [2010]). It is not clear how well these formulations could support the more complex resampling issues arising in continuum elasticity.

There is a huge literature on adaptive meshing to improve the accuracy of finite element methods. See Oden and Demkowicz [1989] for a numerical survey, and Jones and Plassmann [1997] for a geometric survey, of hierarchical mesh refinement, also known as *h-adaptivity*. See Budd, Huang, and Russell [2009] for a survey of moving mesh methods, also known as *r-adaptivity*. Examples of mesh refinement applied to problems in graphics include Shamir *et al.* [2000], Ganovelli *et al.* [2001], Debunne *et al.* [2001], Grinspun *et al.* [2002], and Capell *et al.* [2002]. Adaptive methods are a specialized form of dynamic meshing, and they are effective for adding detail only where needed, but they have the drawback that the deformation field is always anchored to the original coarse mesh. They do not suffice to maintain high quality in a mesh undergoing gross plastic flow, fracture, cutting, or other phenomena that fundamentally reshape the simulation domain.

There are few examples of dynamic meshing that stretch beyond *h*- and *r*-adaptivity. The most notable is the ballistic penetration simulation of Mauch *et al.* [2006]. Unfortunately, they provide little detail about their algorithms for remeshing and resampling, and no data on tetrahedron quality. An especially notable two-dimensional example is the dynamic meshing procedure of Cardoze et al. [2004] for the simulation of circulating blood and the deforming blood cells transported by it.

Specialized remeshing has been used in graphics to address particular phenomena such as fracture [O'Brien and Hodgins 1999;

O'Brien et al. 2002; Molino et al. 2004; Müller and Gross 2004; Müller et al. 2004], cutting [Bielser et al. 1999; Sifakis et al. 2007; Steinemann et al. 2006a; Steinemann et al. 2006b], and needle insertion [Chentanez et al. 2009]. Our dynamic mesher is compatible with all these types of specialized remeshing, and it can improve the quality of the meshes they maintain. To demonstrate this, we have successfully integrated it with a fracture algorithm described by O'Brien and Hodgins [1999].

Our dynamic mesher uses local transformations to conservatively maintain a high-quality material space mesh undergoing plastic flow. There is a substantial literature on local methods for tetrahedral mesh improvement, often called mesh "clean-up." The main ingredients of a mesh improvement algorithm are a set of local transformations, which replace small groups of tetrahedra with other tetrahedra of better quality, and a schedule that searches for opportunities to apply them. Important transformations include stellar flips, the *edge removal* operation proposed by Brière de l'Isle and George [1995], and vertex smoothing—the movement of vertices—whose history begins with simple Laplacian smoothing [Hermann 1976] and proceeds through increasingly sophisticated optimization algorithms [Parthasarathy and Kodiyalam 1991; Canann et al. 1993; Freitag et al. 1995]. Influential mesh improvement schedules include one by Joe [1995] and the *sliver exudation* algorithm developed by Cheng *et al.* [2000] and implemented by Edelsbrunner and Guoy [2001].

The transformation schedule most influential to our work is by Freitag and Ollivier-Gooch [1997], who combine topological transformations with a nonsmooth optimization algorithm for vertex smoothing by Freitag, Jones, and Plassmann [1995]. They present a schedule that eliminates most poorly shaped tetrahedra, and they offer empirical recommendations about what makes some schedules better than others. This work was extended by Klingner and Shewchuk [2007], whose most notable addition is a local transformation that inserts (and sometimes deletes) vertices. Their additions make mesh improvement much more reliable: instead of removing most bad tetrahedra, one can now generally remove them all. The authors report that in their test meshes, no dihedral angle is smaller than 31° or larger than 149°.

The reliability of these methods is what makes our use of dynamic meshing possible. Our mesher performs all of the transformations from the papers by Freitag and Ollivier-Gooch and Klingner and Shewchuk, and adds several more ideas described in Section 4 and by Klingner [2009], including an edge contraction operation and a method that uses quadric errors to help smooth vertices on the surface of a curved domain.

## 3 Elastoplastic Deformation Model

We use a linear co-rotational finite element formulation that has become a standard in computer graphics [Irving et al. 2004; Müller and Gross 2004], including the established extensions for plasticity and fracture. For a full treatment of the topic, see Cook *et al.* [2001], or the introduction by Nealen *et al.* [2006] to deformable models in computer graphics. We recount just enough of it to discuss plasticity.

It is useful to think of there being two separate meshes: one in material space and one in world space—although both meshes have the same topology, as illustrated in Figure 3. Let $\mathbf{u}$ be the vector of material-space positions (one for each node), and let $\mathbf{x}$ be the vector of world-space positions. For a tetrahedron whose vertices have indices $i$, $j$, $k$, $\ell$, the $3 \times 3$ shape matrix $\mathbf{X}_m = [\mathbf{u}_j - \mathbf{u}_i \;\; \mathbf{u}_k - \mathbf{u}_i \;\; \mathbf{u}_\ell - \mathbf{u}_i]$ maps barycentric coordinates defined on the tetrahedron to a corresponding vector (relative to $\mathbf{u}_i$) in material space, and the $3 \times 3$ shape matrix $\mathbf{X}_w = [\mathbf{x}_j - \mathbf{x}_i \;\; \mathbf{x}_k - \mathbf{x}_i \;\; \mathbf{x}_\ell - \mathbf{x}_i]$ maps barycentric coordinates to a corresponding vector (relative to $\mathbf{x}_i$) in world space. Therefore, the *deformation gradient*

$$\mathbf{F} = \mathbf{X}_w \mathbf{X}_m^{-1} \tag{1}$$

maps the tetrahedron from material space to world space. Collecting these maps over all the tetrahedra induces a piecewise linear map for the whole domain, as illustrated.

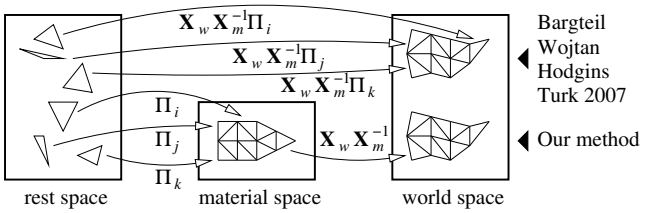



**Figure 3:** *We maintain a material space mesh and its mapping to world space. Remeshing is done in material space. The material space mesh is the rest configuration of the domain when no external force is applied. Plasticity can introduce internal strains so that the rest shape of an isolated tetrahedron does not match its material space shape. Thus, we imagine that each tetrahedron has its own rest space and a plastic offset map* $\mathbf{\Pi}$ *to material space.*

Each deformation gradient can be factored into two parts: a rotation (or reflection) and a matrix capturing how the tetrahedron is stretched or squashed. Following Irving *et al.* [2004], we compute the singular value decomposition $\mathbf{F} = \mathbf{USV}^\mathsf{T}$, where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal (rotations or reflections) and $\mathbf{S}$ is diagonal with all its diagonal entries positive. Observe that $\mathbf{F} = (\mathbf{UV}^\mathsf{T})(\mathbf{VSV}^\mathsf{T})$; the first parenthesized part is an orthogonal matrix, and the second is symmetric and positive definite. The *linearized strain* is the matrix $\epsilon = \mathbf{V}(\mathbf{S} - \mathbf{I})\mathbf{V}^\mathsf{T}$, which represents the deviation of $\mathbf{S}$ from the identity (unstretched) state, in an appropriate coordinate frame.

With an isotropic Hookean constitutive relation $\mathbf{C}$, we compute the first Piola–Kirchhoff stress, namely the matrix $\sigma = \mathbf{UV}^\mathsf{T}\mathbf{C}\epsilon$. Then we compute the elastic forces on the nodes, $\mathbf{f} = \nabla \cdot \sigma$. With these forces, one can derive a standard finite element formulation whose stiffness matrix $\mathbf{K}$ is the Jacobian of the elastic force vector $\mathbf{f}$ with respect to the world coordinates $\mathbf{x}$. The discretized equation of force equilibrium is $\mathbf{M}\ddot{\mathbf{x}} + \mathbf{D}(\mathbf{x})\,\dot{\mathbf{x}} + \mathbf{K}(\mathbf{x})\,\mathbf{x} = \mathbf{f}_{\text{ext}}$, where $\mathbf{f}_{\text{ext}}$ is a vector of external forces, $\mathbf{M}$ is the diagonal (lumped) mass matrix, $\mathbf{D}$ is a nonsymmetric matrix containing velocity-dependent damping terms, and $\mathbf{K}$ is the symmetric stiffness matrix. Both $\mathbf{D}$ and $\mathbf{K}$ vary with the world coordinates $\mathbf{x}$ and must be recomputed each timestep. See Irving et al. for details.

To advance simulation timesteps, we use Newmark time integration for less stiff scenarios and implicit Euler integration for stiffer ones involving collision.

### 3.1 Plasticity

Purely elastic materials are rare in the real world, making plasticity an important ingredient of appealing animations. Unfortunately, the most widely used method for integrating plastic effects into an FEM simulation [O'Brien et al. 2002] is numerically unstable when the material undergoes large plastic deformations. This numerical instability can be avoided with a multiplicative model [Irving et al. 2004; Müller and Gross 2004], but without remeshing these methods can only handle limited amount of plastic flow.

Plastic deformations change the rest shape of an object, so it is natural to change the object's mesh as well. Recently proposed approaches that can simulate extreme plastic deformations [Bargteil et al. 2007; Wojtan and Turk 2008; Wojtan et al. 2009] discretize the world space, and store each element's rest shape implicitly by

storing its deformation gradient, as illustrated in Figure 3. Deformation gradients are repeatedly updated with multiplicative plastic offsets (changing the matrix $\mathbf{\Pi}$ in the illustration). When the deformations threaten to become ill-conditioned, the world-space domain is remeshed from scratch and the deformation gradients are interpolated onto the new mesh. These methods can robustly simulate very plastic materials, but it is not possible to reduce the plasticity below a certain threshold: whenever the deformation gradient is transferred to a new mesh, artificial diffusion introduced by resampling erases some information about the rest state.

As Bargteil *et al.* [2007] note, plastic deformations usually introduce internal strains that persist even when the domain is at equilibrium with no external force applied. The usual, strain-free material space configuration can no longer be embedded in three-dimensional Euclidean space. Bargteil *et al.* respond by discarding material space entirely. Our response is different. Like Bargteil *et al.* [2007], we use multiplicative deformation offset maps to keep track of plastic deformation. These offsets map each element's rest shape to its shape in material space, and are denoted by $\mathbf{\Pi}$ in Figure 3. (The plastic offsets, like the strains, constitute a piecewise constant field over the mesh.) However, we do not allow all the plastic flow to accumulate in these offset maps, because the larger they become, the greater the error when remeshing forces them to be resampled. Instead, at each timestep we update the material-space mesh to its equilibrium shape, which minimizes the total elastic energy—and therefore, the internal strains. Thus the mesh geometry reflects as much of the plastic deformation as possible. Only the non-embeddable portion of the plastic deformation, which induces the internal strains, need be stored in plastic offset maps. The domain shape is much less changed by remeshing than the offset maps are, so this approach reduces artificial diffusion dramatically.

To ensure that we can always return to the true equilibrium shape of the material, we explicitly store that shape as a mesh in material space. We also maintain a world space mesh to represent the displacements and to enable rendering, collision detection, and mesh refinement that is responsive to geometry in world space. These two meshes have the same nodes and topology, but different nodal positions. They must have high quality in material space, and no inverted tetrahedra, at the very least, in world space. Thus, we remesh before the mesh quality degrades too much. After remeshing, quantities stored on the mesh are resampled onto the new mesh, inevitably introducing interpolation errors. Because this error accumulates over time, we remesh as conservatively as possible: instead of remeshing the whole domain, we locally repair bad tetrahedra. The combination of relaxing the plastic offsets and conservative local remeshing in a material space mesh is what allows us to handle the full spectrum of materials from fully elastic to extremely plastic.

After each timestep, we compute the plastic flow from the current strain. We redefine the deformation gradient to be

$$\mathbf{F} = \mathbf{X}_w \mathbf{X}_m^{-1} \mathbf{\Pi}, \tag{2}$$

the map from a tetrahedron's rest space to world space. To simulate plastic flow, update $\mathbf{\Pi}$ to absorb a portion of the symmetric (non-rotational) part $\mathbf{VSV}^\mathsf{T}$ of the deformation gradient:

$$\mathbf{\Pi} \leftarrow \mathbf{\Pi V} \left( \frac{\mathbf{S}}{(\det \mathbf{S})^{1/3}} \right)^{-\gamma} \mathbf{V}^\mathsf{T}, \text{ where } \gamma = \Delta t \, \nu \frac{\|\sigma\| - \tau}{\|\sigma\|}. \tag{3}$$

$\gamma$ determines how much of the deformation is absorbed in a timestep $\Delta t$, in terms of the plastic yield threshold $\tau$, the plastic flow rate $\nu$, and the Frobenius norm of the stress tensor. We enforce $\gamma \in [0, 1]$. Plastic flow changes a tetrahedron's shape in rest space, but its rest volume is preserved, because the matrices multiplied by $\mathbf{\Pi}$ all have determinant 1.

To implement work hardening or softening, we increase the plastic yield $\tau$ by $\kappa\gamma\|\sigma\|$ after each plastic update, where $\kappa$ determines the amount of work hardening (if positive) or softening (if negative).

After the plastic offset maps are updated, we relax the material space mesh to its equilibrium shape. The equilibrium is at an energy minimum, and we find the material space positions $\mathbf{u}'$ that minimize the strain energy of the material space mesh,

$$\mathbf{u}' = \operatorname*{argmin}_{\mathbf{u}} \sum_i V_i \boldsymbol{\epsilon}_i(\mathbf{u}) \, \mathbf{C} \boldsymbol{\epsilon}_i(\mathbf{u}), \tag{4}$$

where $V_i$ is the volume of element $i$ and $\boldsymbol{\epsilon}_i$ is its strain matrix, written as a function of the displacement vector $\mathbf{u}$. This is a nonlinear optimization problem, which we solve with a simple quasi-Newton method. Because the plastic offsets change only moderately between timesteps, the method converges quickly.

This step yields updated world coordinates $\mathbf{u}'$. We adjust the plastic offsets to reflect the change:

$$\mathbf{\Pi} \leftarrow \mathbf{X}'_m \mathbf{X}_m^{-1} \mathbf{\Pi}, \tag{5}$$

where $\mathbf{X}'_m = [\mathbf{u}'_j - \mathbf{u}'_i \;\; \mathbf{u}'_k - \mathbf{u}'_i \;\; \mathbf{u}'_\ell - \mathbf{u}'_i]$. This transformation preserves exactly the shape of each tetrahedron in rest space. It involves no resampling or interpolation—it is accurate to machine precision.

Both plastic flow and our relaxation of the material space mesh can change the volumes of tetrahedra in material and world space, but tetrahedron volumes are always invariant in rest space. Both phenomena deform and often degrade the tetrahedra in material space. The accumulated deformation over many timesteps eventually necessitates remeshing.

# 4 Dynamic Mesh Improvement

The accuracy of our simulations depends on the shapes and sizes of the tetrahedral elements that comprise the mesh in material space. If these elements become sufficiently degraded by plastic flow, the simulation cannot be trusted; if they become inverted, the simulation may be completely nonsensical. To maintain high tetrahedron quality and control the tetrahedron sizes, we use mesh improvement software that conservatively remeshes small portions of the mesh, changing as little as possible during each timestep and thus limiting artificial diffusion.

What makes this approach possible is recent algorithms for mesh improvement that are substantially more reliable than previous methods. Traditional mesh generation methods are not suitable for this purpose, partly because it is difficult to determine how large a region to remesh (it is rarely possible to replace *just* the bad tetrahedra), and partly because most mesh generation algorithms introduce new nonconforming vertices on the boundary of the remeshed region (*e. g.* Delaunay methods) or do not reliably create tetrahedra of uniformly high quality (*e. g.* advancing front methods). Instead, we use hill-climbing optimization to apply local mesh transformations.

## 4.1 Mesh Improvement by Hill Climbing

The heart of our dynamic mesher is a *hill-climbing* method that chooses one of the operations described in Section 4.2 and considers applying it to a specific site in the mesh. An operation is applied only if the quality of the changed mesh will be greater than that of the unchanged mesh. Successive operations monotonically improve the mesh, so the final mesh cannot be worse than the input mesh. Hill climbing stops when the quality of every tetrahedron is above some threshold $q_{\min}$, or when further optimization promises too little gain for too much expenditure of time.

The objective function by which we judge a mesh is its *quality vector*: a vector listing a numerical rating of quality for each tetrahedron, ordered from worst to best. Two meshes' quality vectors are compared *lexicographically* so that, for instance, an improvement in the second-worst tetrahedron improves the objective value even if the worst tetrahedron is not changed. A nice property of the quality

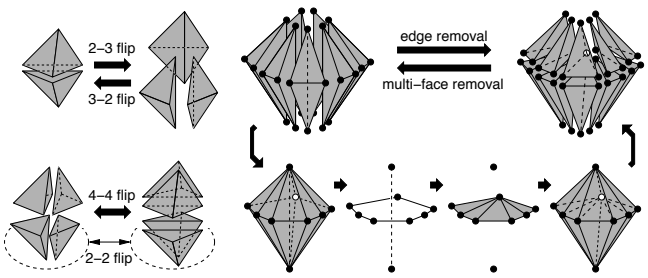



**Figure 4:** *Examples of topological transformations.*

vector is that if an operation replaces a small subset of tetrahedra in a mesh with new ones, we only need to compare the quality vectors of the submeshes constituting the changed tetrahedra (before and after the operation). If the submesh improves, the quality vector of the whole mesh improves.

There is a large literature on *quality measures* that assign each tetrahedron a numerical quality. An excellent measure is the *volume-length measure* suggested by Parthasarathy, Graichen, and Hathaway [1994], denoted $V/\ell_{\text{rms}}^3$, which is the signed volume of a tetrahedron divided by the cube of its root-mean-squared edge length. We find it to be fast and effective as both a quality measure and an objective function for optimization-based smoothing. For dynamic meshing, however, we obtain better results if we modify the quality measure to be even less forgiving of tetrahedra that have an unduly short edge. We achieve this with the quality measure

$$6\sqrt{2}V\frac{\ell_{\text{harm}}}{\ell_{\text{rms}}^4}, \tag{6}$$

where $\ell_{\text{harm}}$ is the harmonic mean of the tetrahedron's six edge lengths. Tetrahedron quality ranges from zero for a degenerate tetrahedron (whose four vertices are coplanar) to a maximum of one for an equilateral tetrahedron. In Section 4.6, we extend this quality measure to circumstances where we desire anisotropic tetrahedra.

## 4.2 Mesh Operations

For a dynamic mesher to achieve consistently good quality timestep after timestep, it must employ a large variety of local mesh improvement operations to repair poorly shaped tetrahedra and to locally refine or coarsen the material space mesh to match the gradient of the strain field. *Smoothing* is the act of moving a vertex to improve the quality of the elements adjoining it. Smoothing does not change the topology (connectivity) of the mesh. *Topological transformations* are operations that change the mesh topology by removing elements from a mesh and replacing them with a different set of elements occupying the same space. Examples appear in Figure 4, including 2-3 flips, 3-2 flips, 4-4 flips, and 2-2 flips. The numbers denote the number of tetrahedra removed and created, respectively.

To smooth vertices, we use a nonsmooth optimization algorithm of Freitag, Jones, and Plassmann [1995] that can optimize the worst tetrahedron in a group—for instance, maximizing the minimum dihedral angle among the tetrahedra that share a specified vertex. Vertices on the boundary of the mesh require special treatment, described in Section 4.3, to limit changes to the shape of the surface.

Some topological transformations are more complicated than the basic flips. *Edge removal* [Briere de l'Isle and George 1995] is a transformation that removes a single edge from the mesh, along with all the tetrahedra that include it. It includes the 3-2 and 4-4 flips, but more generally replaces $m$ tetrahedra with $2m - 4$; Figure 4 (right) illustrates replacing seven tetrahedra with ten. The tetrahedra sharing the removed edge are replaced by other tetrahedra chosen to maximize the quality of the worst new tetrahedron. This choice can be efficiently made by a dynamic programming algorithm of Klincsek [1980].

*Multi-face removal* [de Cougny and Shephard 1995] is the inverse of edge removal, and includes the 2-3 and 4-4 flips. An $m$-face removal replaces $2m$ tetrahedra with $m + 2$. We use an optimization algorithm of Shewchuk [2002] to find the optimal multi-face removal operation to target a specified triangular face.

*Vertex insertion* is a transformation with two uses. We rely upon it to refine the mesh wherever we require higher resolution, and to eliminate stubborn tetrahedra of poor quality. Our vertex insertion algorithm is akin to Delaunay vertex insertion: it hollows out a polyhedral cavity by deleting selected tetrahedra, and replaces them with new tetrahedra that each join the new vertex to a face of the cavity. This operation differs from Delaunay vertex insertion in several ways: it decides which tetrahedra to delete not with the circumsphere criterion, but rather with a combinatorial optimization algorithm that maximizes the quality of the worst new tetrahedron; and it does not always increase the number of vertices in the mesh, because it sometimes deletes vertices by deleting all their incident tetrahedra. In practice, vertex insertion is consistently effective only as part of a *compound operation*: after a vertex insertion, our mesher attempts edge and multi-face removal operations on the new tetrahedra, and smoothing of all their vertices, before deciding whether to accept or roll back the vertex insertion. See Klingner and Shewchuk [2007] for details.

*Edge contraction* also has two uses: to coarsen the mesh where its tetrahedra are unnecessarily small, and to remove tetrahedra that have poor quality because an edge is too short. A contraction operation removes an edge from the mesh, replacing its two endpoints with a single vertex. The tetrahedra that share the contracted edge are deleted from the mesh. The location of the contracted vertex is determined by optimization-based smoothing (taking into account the quadrics discussed in Section 4.3, so that boundary vertices stay on the boundary). An edge contraction operation is rejected if it worsens the mesh quality, if it changes the shape of the domain too much (see Section 4.4), or if it changes the topology of the domain boundary—for example, it is forbidden to contract an edge that connects two boundary vertices through the domain interior.

## 4.3 Quadric Smoothing of Surface Vertices

To maintain high quality in a mesh undergoing gross deformations, we must smooth the vertices on the surface of the mesh, not just the interior ones. It is not possible to have high-quality tetrahedra if the boundary triangles have poor quality. But if the boundary is not flat, moving a surface vertex changes the shape of the domain and might fail to preserve mass. Moreover, it is not clear how to smooth a vertex along a surface that in principle should be curved, but for which we know only a piecewise linear approximation.

To strike a balance between mesh quality and shape preservation, we introduce *quadric smoothing*, which employs for each surface vertex a well-known measure of surface shape error that Garland and Heckbert [1997] call the *quadric error*. This measure is sometimes used to evaluate the error on dynamically remeshed surfaces [Jiao 2007; Brochu and Bridson 2009].

Suppose that a dynamic meshing algorithm displaces a vertex $v$ on the boundary of a tetrahedral mesh from the position it had at the beginning of the timestep. Consider the triangular faces that adjoin $v$ and lie on the boundary of the mesh (ignoring interior faces), in their original positions at the beginning of the timestep (before $v$ was displaced). Each face induces a plane $\{\mathbf{x} : \mathbf{n}_i^{\mathsf{T}}\mathbf{x} + \delta_i = 0\}$, where $\mathbf{n}_i$ is a unit vector normal to the plane, $\delta_i$ is a scalar offset, and $i$ is

the index of the face and the planes it induces. If $v$ is displaced to the position $\mathbf{x}$, we define its quadric error to be

$$Q(\mathbf{x}) = \sum_i \frac{d_i(\mathbf{x})^2}{a_i^2} = \mathbf{x}^\mathsf{T}\left(\sum_i \frac{\mathbf{n}_i\mathbf{n}_i^\mathsf{T}}{a_i^2}\right)\mathbf{x} + \left(\sum_i \frac{2\delta_i\mathbf{n}_i^\mathsf{T}}{a_i^2}\right)\mathbf{x} + \sum_i \frac{\delta_i^2}{a_i^2},$$

where $d_i(\mathbf{x})$ is the distance from $\mathbf{x}$ to plane $i$ and $a_i$ is the original altitude (pre-displacement) of $v$ in triangle $i$. This definition reflects the fact that a displacement of $v$ rotates the triangle's normal vector by an angle approximately proportional to $d_i(\mathbf{x})/a_i$. $Q(\mathbf{x})$ is quadratic and takes its minimum value (zero) at $v$'s original position.

Quadrics permit us to smooth surface vertices while controlling how much error is introduced into the domain shape. Our vertex smoothing algorithm uses nonsmooth optimization to trade each surface vertex's quadric error against the quality of the adjoining tetrahedra, so a vertex is permitted to move further if some tetrahedron improves dramatically. If the surface is locally nearly flat, the vertex has much freedom to move along the surface, but little to move orthogonally. Vertex displacement and tetrahedron quality is admittedly an apples-to-oranges comparison, but we find that the easiest and most effective way to incorporate vertex displacements into optimization-based smoothing is to assign each surface vertex a quality of $q(\mathbf{x}) = \alpha - \beta Q(\mathbf{x})$, where $\alpha$ is an offset parameter and $\beta$ is a scale parameter, and compare it directly against tetrahedron quality, which ranges from zero to one.

Our smoothing algorithm relocates each internal vertex so as to maximize the quality of the worst adjoining tetrahedron. For surface vertices, it maximizes the minimum of the adjoining tetrahedra *and* the quality of the vertex itself. The nonsmooth optimization algorithm of Freitag et al. [1995] accommodates this notion of surface vertex quality with virtually no change.

The default values in our implementation are $\alpha = 0.8$ and $\beta = 1,200$. With these parameters, the changes in the surface shape are barely perceptible, while mesh quality gets a big boost. The scale parameter $\beta$ controls how quickly a vertex is penalized as it moves away from its original position. A tetrahedron must have a quality below the offset parameter $\alpha = 0.8$ to justify moving a surface vertex. If no tetrahedron incident to a surface vertex has a quality less than 0.8, the surface vertex has the lowest quality, and smoothing will move it toward its original position.

Collectively, the quadrics provide a memory of the original domain shape (at the beginning of the timestep). We observe that when a surface vertex is smoothed in pursuit of better tetrahedron quality, the worst incident tetrahedron is often improved by subsequent topological changes that would not otherwise have been possible, which in turn permits a subsequent smoothing step to move the vertex back to its original position, or at least closer.

We recompute the quadrics from scratch after each simulation timestep. Quadrics do not persist from timestep to timestep, but they do persist through all the dynamic meshing passes performed during any single timestep. A vertex insertion operation sometimes creates a new vertex on a face or edge on the mesh surface. We compute a quadric for it by considering the faces it is inserted on with their vertices repositioned where they were at the *beginning* of the timestep. That way, if those vertices revert to their original positions, the new vertex will also tend to revert to an appropriate position.

### 4.4 Other Operations that Modify the Mesh Surface

It is crucial to include operations that change the topology of the mesh boundary, as good tetrahedra are impossible without good boundary triangles. Unfortunately, these operations usually change the shape of the domain, so we must limit the amount of change.

An important special case of edge removal is the 2-2 flip, which has the effect of flipping an edge on the surface of the mesh. If the two flipped boundary triangles are not coplanar, the flip reshapes the domain. We permit a 2-2 flip only if the total volume of the two tetrahedra created by the flip differs from the volume of the two deleted tetrahedra by less than 9%, and the surface normal vectors of the two boundary triangles change by less than 8°. If this sounds too permissive, our experience shows that setting these parameters too low causes evolving surfaces to slowly deteriorate and wrinkle.

We use quadrics to judge when it is possible to contract an edge on the mesh boundary without distorting the domain shape too much. An edge contraction is permitted only if one endpoint of the edge can move to the same position as the other endpoint without the quality of the moved vertex falling below the threshold $q_{\min}$.

### 4.5 A Dynamic Mesh Improvement Schedule

Whereas standard mesh improvement algorithms try to improve the mesh to as high a quality as possible, changing as much of the mesh as necessary, a dynamic mesher must remesh conservatively to limit artificial diffusion. Our dynamic mesh improvement schedule acts only when some tetrahedron falls below a minimum threshold for quality, and it attempts to fix it while changing as few tetrahedra as possible. To fix a bad tetrahedron, it first tries the most local transformations (like 2-3 and 3-2 flips), and progresses only if necessary to the most disruptive ones (like smoothing).

Listing 1 lists pseudocode for our dynamic improvement schedule. For each tetrahedron in a mesh whose quality is worse than some specified minimum quality $q_{\min}$, the schedule invokes the procedure IMPROVETET to try to improve it. IMPROVETET maintains a set $A$ of tetrahedra that were touched during hill climbing. Initially $A$ contains just a single bad tetrahedron, but as IMPROVETET works, it adds to $A$ all the tetrahedra it creates or modifies.

IMPROVETET iterates up to ten times, each time working through four passes of mesh improvement: edge removal and face removal operations, edge contractions, vertex insertions, and smoothing. Each mesh improvement pass maintains a set $A$ of tetrahedra, which includes all the new tetrahedra created by the pass, all the tetrahedra that have had a vertex smoothed by the pass, and all the surviving tetrahedra that were previously in $A$. The union of the tetrahedra in $A$ is a connected region that tends to grow as mesh transformations occur. The passes try to improve all the tetrahedra in $A$—not just those whose quality is below $q_{\min}$—because experience shows that a bad tetrahedron often cannot be eliminated until its neighbors improve. On rare occasions, we see a single tetrahedron that resists improvement for six or more iterations of the outer loop, when $A$ has grown to include hundreds or thousands of nearby tetrahedra.

The code for EDGECONTRACTIONPASS and the smoothing pass is included here. TOPOLOGICALPASS and INSERTIONPASS are omitted because they are essentially the same passes described by Klingner and Shewchuk [2007]. INSERTIONPASS is the most complicated pass; it follows each vertex insertion with local edge removals, multi-face removals, and vertex smoothing before judging whether the insertion is successful or must be rolled back. TOPOLOGICALPASS is similar in character to EDGECONTRACTIONPASS: it tries to remove every edge, then every face, of the tetrahedra in $A$. (As usual, transformations that do not improve the quality vector are rejected.) Because topological transformations can bring additional tetrahedra into $A$, it is possible that the minimum quality of $A$ is lower after a pass than it was before. To keep $A$ from growing too much due to repeated runs of TOPOLOGICALPASS, we add a check to the end of TOPOLOGICALPASS that rolls back the entire pass if the minimum quality has worsened. No such check is performed for the other improvement passes.

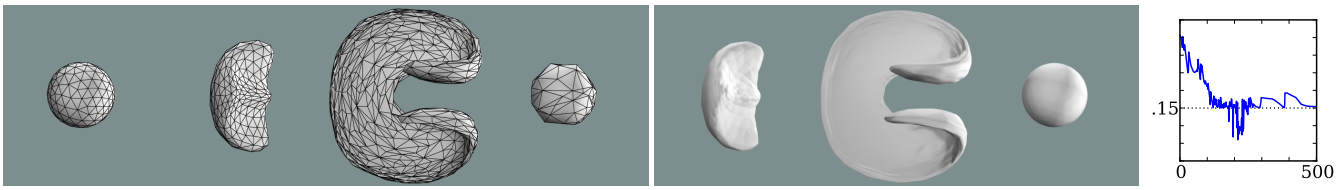



**Figure 5:** *The Enright test. A prescribed velocity field stretches a purely elastic sphere, which snaps back to its original shape after the constraints are released. Four frames of the simulation mesh, three frames of an embedded surface (the initial surface is omitted, as it is identical to the final shape), and a graph of the quality of the worst tetrahedron as a function of timestep number.*

```
EDGECONTRACTIONPASS(A, M)
{ A is a subset of the tetrahedra in the mesh M }
1   E ← set of all edges of tetrahedra in A
2   for each edge e ∈ E
3       if e still exists
4           Attempt to contract edge e and smooth the vertex
                thus created with nonsmooth optimization.
            { See Section 4.2 for ways the attempt can fail. }
5   return set containing the surviving tetrahedra in A and the
                tetrahedra in M altered by edge contractions.

IMPROVETET(M, t, q_min)
{ t is a tetrahedron in the mesh M }
{ q_min is the minimum acceptable tetrahedron quality }
6   A ← { t }
7   for i ← 1 to 10
8       do
9           A ← TOPOLOGICALPASS(A, M)
10          if the quality of the worst tetrahedron in A ≥ q_min
11              return
12      while A is changed by the topological pass
13      A ← EDGECONTRACTIONPASS(A, M)
14      if the quality of the worst tetrahedron in A ≥ q_min
15          return
16      A ← INSERTIONPASS(A, M)
17      if the quality of the worst tetrahedron in A ≥ q_min
18          return
        { smoothing pass begins here }
19      V ← set of all vertices of the tetrahedra in A
20      for each vertex v in V
21          Smooth v with nonsmooth optimization.
22          A ← A ∪ the tetrahedra adjoining v in M
23      if the quality of the worst tetrahedron in A ≥ q_min
24          return

DYNAMICIMPROVEMESH(M, q_min)
25  B ← set of tetrahedra in M with quality less than q_min
26  for each tetrahedron t ∈ B
27      if t still exists and has quality less than q_min
28          IMPROVETET(M, t, q_min)
```

**Listing 1:** *The dynamic mesh improvement schedule.*

Some passes change more tetrahedra than others. The topological pass is the most conservative: flips, edge removal operations, and multi-face removal operations typically change only a few tetrahedra. The edge contraction pass is worse: it adds to $A$ all the tetrahedra incident to the endpoints of each contracted edge. The insertion pass is worse still because vertex insertion is a compound operation. The smoothing pass is the worst of all, because $A$ expands everywhere outward by an entire layer of tetrahedra. Every tetrahedron incident to a smoothed vertex is included in $A$. IMPROVETET thus performs the passes in this order, and terminates as soon as the worst tetrahedron in $A$ has a quality of at least $q_{\min}$.

It is counterintuitive that the insertion pass is less disruptive than the smoothing pass. After all, the insertion pass itself smooths vertices as part of each compound vertex insertion operation. In practice, though, a smoothing pass rarely brings a bad tetrahedron above the minimum quality threshold; if smoothing can do it at all, it usually takes multiple passes, each of which enlarges $A$. In contrast, vertex insertion (and subsequent cavity improvement) can often surgically remove a bad tetrahedron in one attempt. Our experience is that putting the smoothing pass before the vertex insertion pass leads to more remeshing.

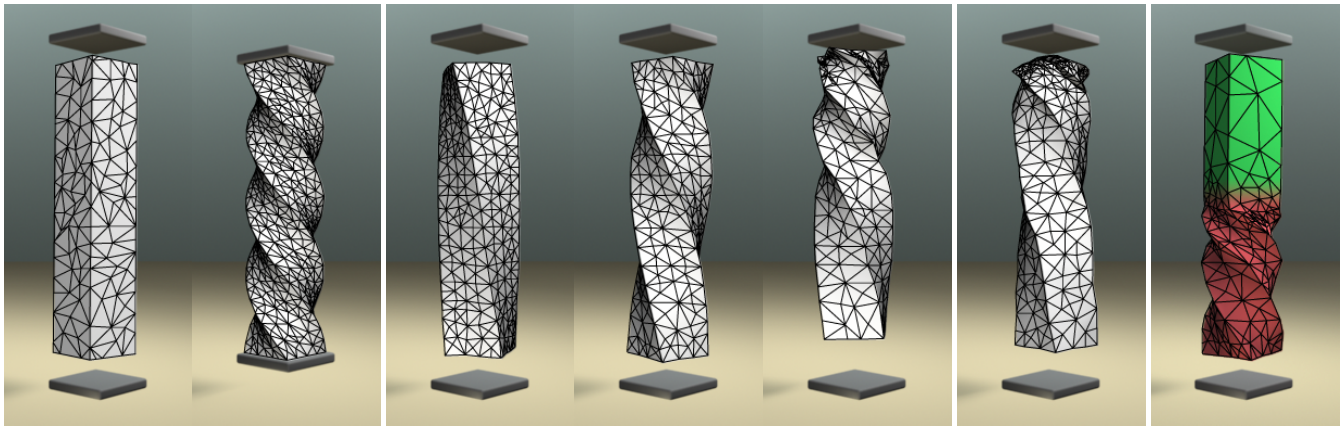
Because it is so conservative, TOPOLOGICALPASS is the only improvement pass in the dynamic schedule that is permitted to run repeatedly, as long as it makes progress in improving $A$. Experience shows that it is not wise to iterate any other pass more than once without trying the other passes as well, because it is common that $A$ contains a bad tetrahedron that is easily removed by one pass but not by the others. We tried a variety of schedules for ordering and iterating the passes before settling on the listed IMPROVETET as the schedule that performed the least remeshing.

### 4.6 Refinement, Coarsening, and Anisotropy

Our dynamic mesher includes algorithms for local refinement and coarsening, so that we can adaptively refine a mesh in regions where the need for accuracy is great, coarsen it in regions where the need has passed, and even demand anisotropic tetrahedra in regions where they are advantageous. Our simulations tailor tetrahedron sizes and anisotropy to reflect the gradient of the strain (so the strain field is accurately represented) or the gradient of the displacement (so the geometry is accurately represented).

The simulation tells the mesher its desires through a *sizing field* that specifies the ideal edge length at each point in material space. Although equilateral tetrahedra are usually considered ideal, anisotropic tetrahedra with the right eccentricities and orientations are often more efficient and accurate in simulations that have anisotropic physical behavior. For these scenarios, the sizing field is a metric, represented by a $3 \times 3$ symmetric positive definite tensor field $\mathbf{M}(\mathbf{x})$. The ideal tetrahedron is one that, under this metric, is equilateral with edge lengths of 1. Let the *deformation tensor* $\mathbf{M}^{1/2}(\mathbf{x})$ be the symmetric positive definite square root of the metric tensor $\mathbf{M}(\mathbf{x})$. Given a tetrahedron in material space, the mesher judges its shape by first applying the affine transformation $\mathbf{M}^{1/2}(\mathbf{x})$ (to account for the desired anisotropy and scale), then computing the quality measure (6) for the transformed tetrahedron. Thus, its quality is measured in the metric $\mathbf{M}(\mathbf{x})$. If this quality falls below a threshold $q_{\min}$, our dynamic mesher tries to repair it. With this simple change, the mesher can create meshes with almost any desired anisotropy, so long as the metric tensor field is sufficiently smooth.

The mesher judges a tetrahedron's size by checking that the edge lengths of the transformed tetrahedron are sufficiently close to 1. If they are too long, the mesher refines locally; if they are too short, it tries to coarsen. When our dynamic mesher is invoked, it runs a *size control phase* prior to the mesh improvement schedule of Section 4.5. The size control phase uses the same mesh operations, but it is permitted to worsen the quality of the mesh, with the expectation that it will be repaired during the improvement phase.



(a) (b) (c) (d) (e) (f) (g)

**Figure 6:** *Twisting bars with different plasticity values. (a) Initial rest shape and final shape of a purely elastic bar. (b) Maximum deformation of the simulation. (c)–(e) Rest shapes at the ends of simulations with low, medium, and high plasticity. (f) Final state of a purely elastic simulation that remeshes in world space instead of material space, thereby accumulating artificial plasticity; compare with (a). (g) Final state of a simulation with two different materials: the upper part is purely elastic, the lower part highly plastic.*

Any edge that is too long has a vertex inserted at its midpoint, unless the insertion operation creates an edge that is unacceptably short (in which case it is rolled back). We use the standard vertex insertion operation, modified so that it is not allowed to delete any vertex (thereby inadvertently coarsening the mesh.) Any edge that is too short is contracted, unless doing so would create a degenerate or inverted tetrahedron, or make an excessively large change to the mesh boundary (as discussed in Section 4.4).

How does the simulation choose a metric tensor field? One option is to refine the mesh where the displacement field has a large gradient, thereby improving the geometric accuracy wherever the domain is stretched or twisted, as illustrated in Figure 5. This is accomplished simply by providing the dynamic mesher the metric tensor $\mathbf{M} = \mathbf{F}^{\mathrm{T}}\mathbf{F}$, where $\mathbf{F}$ is the deformation gradient (1), so the mesher evaluates tetrahedra by measuring them in world space. The metric tensor field defined this way is piecewise constant over the mesh tetrahedra, but we smooth it with piecewise linear interpolation (described below).

For most simulations, a desire for accuracy obliges us to replace the deformation gradient with the gradient of the strain (or to make sure the mesh is fine enough to satisfy both). To do so, we must approximate the strain gradient. Given a tetrahedron $s$ with deformation gradient $\mathbf{F}$, let $T$ be the set of tetrahedra that share a vertex with $s$. We assign $s$ the metric tensor

$$\mathbf{M} = \frac{\kappa}{|T|} \sum_{t \in T} \frac{\mathbf{d}_t \mathbf{d}_t^{\mathrm{T}}}{\|\mathbf{d}_t\|_2^2} \frac{\max\left\{\left\|\mathbf{F}\mathbf{F}_t^{-1}\right\|_2, \left\|\mathbf{F}_t\mathbf{F}^{-1}\right\|_2\right\}}{\|\mathbf{d}_t\|_2}, \tag{7}$$

where $\mathbf{d}_t$ is the vector connecting the barycenters of $s$ and $t$, $\mathbf{F}_t$ is the deformation gradient of $t$, and the scalar $\kappa$ determines the desired accuracy. Observe that for any unit vector $\mathbf{d}$, the outer product matrix $\mathbf{d}\mathbf{d}^{\mathrm{T}}$ has eigenvector $\mathbf{d}$ with eigenvalue 1. Thus, the eigenvectors and eigenvalues of $\mathbf{M}$ indicate which gradient directions are most strongly weighted.

$\mathbf{M}$ is often evaluated at positions not clearly associated with any particular tetrahedron. We precompute $\mathbf{M}$ for all tetrahedra of the original mesh, and use spatial hashing to find the closest tetrahedron to a query point. Especially for very anisotropic meshes with varying resolution, nearest neighbor searches dominate the time complexity of the remeshing process. To speed up evaluations of the sizing tensor field, we compute $\mathbf{M}$ on a regular grid and use linear interpolation to rapidly evaluate $\mathbf{M}$ at an arbitrary point in space. Since the lookup uses material space positions, $\mathbf{M}$ is uniquely defined even if the world space mesh is self-intersecting. $\mathbf{M}$ is not well-defined if the material space mesh is self-intersecting. While we do not explicitly prevent self-intersections in our plastic relaxation, it would be straightforward to add a penalty term for self-intersections to the strain energy function (4). We have not encountered the need in our simulations.

## 5 Resampling Physical Properties

After mesh improvement is complete, we have to transfer simulation data from the old mesh to the new, including the world position, velocity, acceleration, and perhaps residual fracture stress tensor at each vertex, and the plastic offset map and (for work hardening and softening) accumulated plastic stress at each tetrahedron. Conservative remeshing pays off here: in most frames, only a small fraction of the mesh vertices and tetrahedra is modified. Data associated with unmodified entities are simply copied to the new mesh.

We use linear interpolation to estimate properties at inserted or smoothed nodes. If a node was smoothed and its new position is not in any old tetrahedron, we extrapolate from the closest tetrahedron. We approximate per-element quantities as averages of intersecting elements in the source mesh, weighted by the intersection volumes.

Resampling the strain field is a major cause of artificial plasticity in methods that rely on the strain field to store information about the rest shapes of the elements. We store plastic offsets that have to be resampled, but because we always relax the material space mesh to its global rest shape, those plastic offsets are small. The plastic offsets are multiplicative, and cannot simply be added or averaged. We therefore resample the strain field onto the new elements, and reconstruct the plastic offsets from the strain. Specifically, we compute the plastic offset $\mathbf{\Pi}$ for an element by locally averaging the Green strain field. Consider an element of volume $V$ that overlaps a set of old elements with overlap volumes $V_i$, where $V = \sum_i V_i$. Like Bargteil *et al.* [2007], we compute the Green strain $\epsilon_i = \mathbf{F}^{\mathrm{T}}\mathbf{F} - \mathbf{I}$ for the old elements, and integrate it over the new element,

$$\epsilon = \frac{1}{V} \sum_i V_i \epsilon_i. \tag{8}$$

We then recover the plastic offset map from (2):

$$\mathbf{\Pi} = \frac{\tilde{\mathbf{\Pi}}}{(\det \tilde{\mathbf{\Pi}})^{1/3}}, \text{ where } \tilde{\mathbf{\Pi}} = \mathbf{X}_m\mathbf{X}_w^{-1}\mathbf{F} = \mathbf{X}_m\mathbf{X}_w^{-1}\sqrt{\epsilon + \mathbf{I}}. \tag{9}$$

| Example | Figure | Tetrahedra (init/max/final/graph) | $t_{\text{Integration}}$ | $t_{\text{Plasticity}}$ | $t_{\text{Remeshing}}$ | $t_{\text{Sizing}}$ | $t_{\text{total}}$ (avg/std dev/graph) | $q_{\text{worst}}$ |
|---|---|---|---|---|---|---|---|---|
| Drip | 1 | 425/4,143/4,143 | 0.01/0.01 | 0.12/0.14 | 0.02/0.18 | 1.27/2.39 | 1.41/2.47 | 0.1500 |
| Masticator | 2 (a) | 1,041/38,211/1,345 | 0.16/0.22 | n/a | 0.10/0.12 | 7.51/9.44 | 7.77/9.77 | 0.1504 |
| | 2 (b) | 1,041/1,041/1,041 | 0.01/0.00 | n/a | n/a | n/a | 0.01/0.00 | n/a |
| | 2 (c) | 1,041/27,235/27,235 | 0.22/0.06 | 0.51/0.63 | 0.19/0.05 | 58.29/22.66 | 59.21/22.85 | 0.1500 |
| Enright | 5 | 528/4,733/379 | 0.01/0.01 | n/a | 103.67/407.17 | | 103.88/408.30 | 0.0601 |
| Bar | 6 (b) | 214/3,543/373 | 0.01/0.00 | n/a | 0.01/0.01 | 0.30/0.38 | 0.31/0.39 | 0.1500 |
| | 6 (c) | 214/2,924/2,924 | 0.01/0.00 | 0.05/0.09 | 0.02/0.01 | 3.42/1.71 | 3.50/1.71 | 0.2806 |
| | 6 (d) | 214/2,233/2,229 | 0.01/0.00 | 0.03/0.01 | 0.02/0.01 | 2.59/1.13 | 2.65/1.14 | 0.1868 |
| | 6 (e) | 214/2,021/2,009 | 0.01/0.00 | 0.03/0.01 | 0.02/0.01 | 2.40/0.92 | 2.46/0.93 | 0.2087 |
| | 6 (g) | 214/2,810/1,455 | 0.01/0.00 | 0.01/0.01 | 0.02/0.60 | 0.89/0.31 | 0.93/0.68 | 0.1440 |
| Fracture | 7 (a) | 4,488/12,081/12,045 | 0.48/0.40 | 1.29/0.97 | 1.43/1.27 | 1.45/0.32 | 4.73/1.99 | 0.1563 |
| | 7 (b) | 4,488/32,621/32,621 | 0.72/0.42 | 1.78/0.89 | 3.48/0.75 | 8.67/1.80 | 15.00/3.17 | 0.2452 |
| | 7 (c) | 4,488/109,262/109,262 | 3.07/0.85 | 1.04/0.34 | 10.25/2.80 | 1.23/0.24 | 26.82/39.15 | 0.1518 |

**Table 1:** *Statistics for simulation examples. The third column lists the initial, maximum, and final number of tetrahedra for each simulation, and graphs the number as a function of the timestep. Subsequent columns show computation times (average/standard deviation) per simulation timestep, measured in seconds on a single core of a 2.93 GHz Intel Xeon. Times appear for time integration (including stiffness matrix computation), plastic relaxation, remeshing, and evaluation of the sizing field, as well as the total running time per timestep (also graphed as a function of the timestep). Total time excludes file I/O and rendering. The final column shows the quality of the worst tetrahedron used for simulation (excluding small fragments in the fracture simulations); compare with the improvement threshold* $q_{\min} = 0.15$.

We compute the square root of a symmetric matrix from its eigendecomposition.

## 6 Elastoplastic Animation Examples

Figure 6 compares animations of six vertical elastic bars. Each bar begins as a relatively coarse mesh, then has its top end twisted for one full rotation in four seconds while its bottom end is held fixed. The mesh is locally refined as dictated by the stress gradient; without this refinement, the mesh resolution would be insufficient to model the twisting. Then both ends of the bar are released. The meshes are coarsened when it is safe for the mesher to do so. The surfaces of the plastic bars are sufficiently deformed that coarsening takes place almost exclusively in their interiors, but the purely elastic bar exhibits coarsening of its surface triangles too. The mesher tries to maintain a minimum tetrahedron quality of $q_{\min} = 0.15$ throughout (and throughout all the other simulations described in this section). Each bar is shown after being released and returned to an equilibrium state.

The leftmost bar is purely elastic, and returns to its original shape. The bars (c), (d), and (e) undergo progressively greater amounts of plastic flow, and do not return to their original shapes. The rightmost bar, (f), is purely elastic like (a), but for comparison we have made one crucial change: we remesh in world space, rather than in material space, though we still remesh conservatively. The final rest state is disastrous, and would have been much worse if we remeshed from scratch each timestep. This example illustrates the great advantage of remeshing in material space. Observe that bar (f) is more deformed than bar (d), showing that it takes a good deal of plastic flow to conceal the effects of artificial diffusion. The method illustrated in (f) is similar to that of Bargteil *et al.* [2007]. However, because this bar is fully elastic, Bargteil *et al.* would never trigger remeshing, thus avoiding strain diffusion; but for the lightly plastic bar (c), artificial plasticity would overwhelm the real plastic flow.

Bar (g) consists of two materials: a fully elastic part and a highly plastic part. Although the lower part is plastically deformed, the elastic part maintains its rest shape perfectly. This example highlights the versatility of our method: both materials can be accommodated in the same simulation.

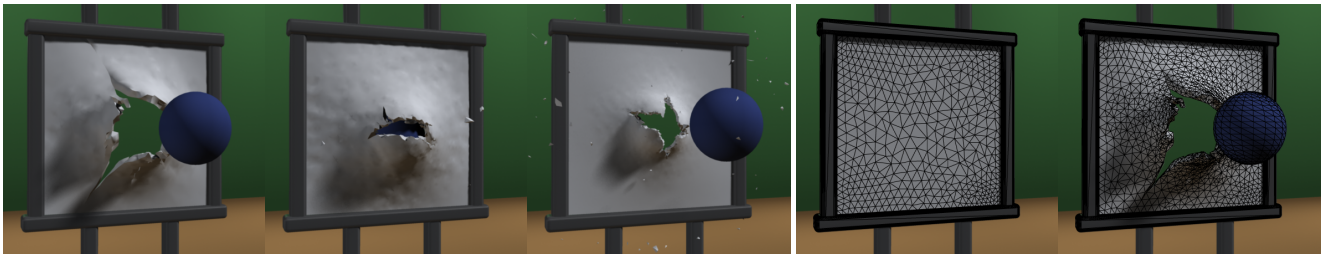
Figure 2 is a similar but more complicated example, which illustrates both a purely elastic and a plastic bar masticated between metal teeth. Observe that the bars are heavily and anisotropically refined (except the center bar, for which we have turned off refinement) but remain coarse at the ends. When the elastic bar returns to its rest shape, the mesh is coarsened. Observe also how failure to refine (center image) yields poor results between the teeth.

Our method is not immune to artificial plasticity, but that problem arises only where we remesh after actual plastic flow has occurred, so we have not been able to visually detect it in any simulation. By contrast, the artificial plasticity caused by world-space remeshing in Figure 6 (f), an example with no actual plasticity, is starkly apparent. It is likely that the plastic example in Figure 2 (rightmost) incurs some artificial plasticity. The purely elastic examples do not, because the plastic offset maps, being the identity, accrue no resampling error during remeshing.

To illustrate how our method copes with an extreme deformation in a purely elastic material, Figure 5 shows the Enright test [Enright et al. 2002], in which a sphere (exhibiting no plastic flow) is deformed by a volume-preserving velocity field; then it snaps back to its original shape after the constraints are released. This test is very difficult for simulations halfway through, when the deformed sphere is extremely thin. Here we use anisotropic tetrahedra—many fewer than would be possible with isotropic tetrahedra alone—and the anisotropic quality measure described in Section 4.6, computed from the deformation gradient. The figure includes a graph plotting the lowest tetrahedron quality as a function of the timestep. The remesher cannot always keep the tetrahedron quality above 0.15. However, the offending tetrahedra are short-lived, and they are removed in subsequent timesteps. The quality never dips below 0.06, posing no threat to the numerical stability of the simulation. The extreme deformation creates lots of badly shaped tetrahedra in every timestep, making this by far our most computationally intensive example.

After the test completes, the original shape is preserved well, except that repeated remeshing has caused the spherical surface to deteriorate somewhat. The three rightmost images in Figure 5 show a fine triangulated surface we have embedded in the mesh, which advects it. The embedded surface is not affected by remeshing, and it ends up being a perfect sphere again.

From Table 1, which tabulates statistics about the mesh sizes and the simulation times for all the examples illustrated in this paper, we make several observations. First, in the purely elastic simulations (Masticator, Bar, Enright), the meshes are aggressively coarsened to nearly their original sizes upon returning to their original shapes. Second, the computation time per timestep is only loosely correlated with the number of tetrahedra. The biggest factor determining the remeshing complexity is the quality of the mesh. Simulations that invalidate many tetrahedra in each timestep, such as



(a) (b) (c) (d) (e)

**Figure 7:** *Adaptive mesh refinement helps a ball to crash through different ductile plates. Frames (d) and (e) show the mesh for the simulation (a) before the projectile penetrates the plate and after the fracture.*

the Enright test, are the most expensive. Third, the simulation time is dominated by the sizing field computation and lookup, in particular nearest-tetrahedron searches. (For the Enright test, we do not use a regular grid to precompute the sizing field, so we give only one figure for sizing and remeshing time in Table 1.) Speeding up evaluations of the sizing field is one of our most pressing needs. A data structure for spatially adaptive storage and evaluation of the sizing field (similar to that of Frisken *et al.* [2000]) could save a huge portion of the simulation cost.

The animation of a fracturing plate in Figure 7 shows that standard fracture simulation methods [O'Brien and Hodgins 1999], which subdivide individual tetrahedra into smaller ones, are compatible with our dynamic mesher. Moreover, we are able to begin the simulation with a relatively coarse mesh; the dynamic mesher refines the regions where the strain gradient becomes high, so that the fracture effects, which require a fine mesh, can proceed accurately. An additional benefit, not apparent from the animation, is that our dynamic mesher repairs most of the poor-quality tetrahedra created when tetrahedra are subdivided, which raises the hope that methods for dynamically changing geometry might soon become reliable enough for engineering simulations requiring high accuracy.

Finally, the dripping viscous fluid in Figure 1 illustrates many of our method's virtues: plastic flow and work hardening; the capability to completely reshape a domain; adaptive mesh refinement that places strongly anisotropic tetrahedra where they are needed at the narrowest part of a tendril; and a final moment of fracture as the drop falls.

The video files for the animations depicted in this paper, and our remeshing software *Pulsar*, are available online[1]. We encourage readers to use Pulsar in their own research.

## Acknowledgments

We thank the other members of the Berkeley Graphics Group for their helpful suggestions. This work was supported in part by NSF Awards CCF-0635381 and IIS-0915462, UC Lab Fees Research Program Grant 09-LR-01-118889-OBRJ, California Discovery Grant COM09S-156646, and by generous support and equipment donations from Intel, NVIDIA, Pixar, Adobe, and Autodesk.

## References

BARGTEIL, A. W., WOJTAN, C., HODGINS, J. K., AND TURK, G. 2007. A finite element method for animating large viscoplastic flow. *ACM Transactions on Graphics 26*, 3, 16:1–16:8.

BIELSER, D., MAIWALD, V. A., AND GROSS, M. H. 1999. Interactive cuts through 3-dimensional soft tissue. *Computer Graphics Forum 18*, 3, 31–38.

[1] http://graphics.berkeley.edu/papers/Wicke-DLR-2010-07

BRIERE DE L'ISLE, E., AND GEORGE, P.-L. 1995. Optimization of tetrahedral meshes. In *Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations*, vol. 75 of *IMA Volumes in Mathematics and its Applications*. 97–128.

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

BUDD, C. J., HUANG, W., AND RUSSELL, R. D. 2009. Adaptivity with moving grids. In *Acta Numerica 2009*, vol. 18. 1–131.

CANANN, S. A., STEPHENSON, M., AND BLACKER, T. 1993. Optismoothing: An optimization-driven approach to mesh smoothing. *Finite Elements in Analysis and Design 13*, 185–190.

CAPELL, S., GREEN, S., CURLESS, B., DUCHAMP, T., AND POPOVIĆ, Z. 2002. A multiresolution framework for dynamic deformations. In *Proc. Symposium on Computer Animation*, 41–48.

CARDOZE, D., CUNHA, A., MILLER, G. L., PHILLIPS, T., AND WALKINGTON, N. 2004. A Bézier-based approach to unstructured moving meshes. In *Proc. Symposium on Computational Geometry*, 310–319.

CARLSON, M., MUCHA, P. J., VAN HORN III, R. B., AND TURK, G. 2002. Melting and flowing. In *Proc. Symposium on Computer Animation*, 167–174.

CARLSON, M., MUCHA, P. J., AND TURK, G. 2004. Rigid fluid: Animating the interplay between rigid bodies and fluid. *ACM Transactions on Graphics 23*, 3, 377–384.

CHENG, S.-W., DEY, T. K., EDELSBRUNNER, H., FACELLO, M. A., AND TENG, S.-H. 2000. Sliver exudation. *Journal of the Association for Computing Machinery 47*, 5, 883–904.

CHENTANEZ, N., FELDMAN, B. E., LABELLE, F., O'BRIEN, J. F., AND SHEWCHUK, J. R. 2007. Liquid simulation on lattice-based tetrahedral meshes. In *Proc. Symposium on Computer Animation*, 219–228.

CHENTANEZ, N., ALTEROVITZ, R., RITCHIE, D., CHO, L., HAUSER, K. K., GOLDBERG, K., SHEWCHUK, J. R., AND O'BRIEN, J. F. 2009. Interactive simulation of surgical needle insertion and steering. *ACM Transactions on Graphics 28*, 3, 88:1–88:10.

COOK, R. D., MALKUS, D. S., PLESHA, M. E., AND WITT, R. J. 2001. *Concepts and Applications of Finite Element Analysis*, fourth ed. John Wiley & Sons, New York.

DE COUGNY, H. L., AND SHEPHARD, M. S. 1995. Refinement, derefinement, and optimization of tetrahedral geometric triangulations in three dimensions. Manuscript.

DEBUNNE, G., DESBRUN, M., CANI, M.-P., AND BARR, A. H. 2001. Dynamic real-time deformations using space & time adaptive sampling. In *Proc. SIGGRAPH 2001*, 31–36.

EDELSBRUNNER, H., AND GUOY, D. 2001. An experimental study of sliver exudation. In *Proc. Tenth International Meshing Roundtable*, 307–316.

ENRIGHT, D., FEDKIW, R., FERZIGER, J., AND MITCHELL, I. 2002. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics 183*, 1, 83–116.

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

FREITAG, L. A., JONES, M., AND PLASSMANN, P. 1995. An efficient parallel algorithm for mesh smoothing. In *Proc. Fourth International Meshing Roundtable*, 47–58.

FRISKEN, S. F., PERRY, R. N., ROCKWOOD, A. P., AND JONES, T. R. 2000. Adaptively sampled distance fields: A general representation of shape for computer graphics. In *Proc. SIGGRAPH 2000*, 249–254.

GANOVELLI, F., CIGNONI, P., MONTANI, C., AND SCOPIGNO, R. 2001. Enabling cuts on multiresolution representation. *The Visual Computer 17*, 5, 274–286.

GARLAND, M., AND HECKBERT, P. 1997. Surface simplification using quadric error metrics. In *Proc. SIGGRAPH '97*, 209–216.

GIBSON, S. F. F., AND MIRTICH, B. 1997. A survey of deformable modeling in computer graphics. Tech. Rep. TR97-17, Mitsubishi Electric Research Laboratories.

GOKTEKIN, T. G., BARGTEIL, A. W., AND O'BRIEN, J. F. 2004. A method for animating viscoelastic fluids. *ACM Transactions on Graphics 23*, 3, 463–468.

GRINSPUN, E., KRYSL, P., AND SCHRÖDER, P. 2002. CHARMS: A simple framework for adaptive simulation. *ACM Transactions on Graphics 21*, 3, 281–290.

HERMANN, L. R. 1976. Laplacian-isoparametric grid generation scheme. *Journal of the Engineering Mechanics Division of the American Society of Civil Engineers 102*, 749–756.

IRVING, G., TERAN, J., AND FEDKIW, R. 2004. Invertible finite elements for robust simulation of large deformation. In *Proc. Symposium on Computer Animation*, 131–140.

JIAO, X. 2007. Face offsetting: A unified approach for explicit moving interfaces. *Journal of Computational Physics 31*, 4, 2472–2493.

JOE, B. 1995. Construction of three-dimensional improved-quality triangulations using local transformations. *SIAM Journal on Scientific Computing 16*, 6, 1292–1307.

JONES, M. T., AND PLASSMANN, P. E. 1997. Adaptive refinement of unstructured finite-element meshes. *Finite Elements in Analysis and Design 25*, 41–60.

KLINCSEK, G. T. 1980. Minimal triangulations of polygonal domains. *Annals of Discrete Mathematics 9*, 121–123.

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

KLINGNER, B. M., FELDMAN, B. E., CHENTANEZ, N., AND O'BRIEN, J. F. 2006. Fluid animation with dynamic meshes. *ACM Transactions on Graphics 25*, 3, 820–825.

KLINGNER, B. M. 2009. *Tetrahedral Mesh Improvement*. PhD thesis, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, California.

KUCHARIK, M., GARIMELLA, R. V., SCHOFIELD, S. P., AND SHASHKOV, M. J. 2010. A comparative study of interface reconstruction methods for multi-material ALE simulations. *Journal of Computational Physics 229*, 7, 2432–2452.

MAUCH, S., NOELS, L., ZHAO, Z., AND RADOVITZKY, R. A. 2006. Lagrangian simulation of penetration environments via mesh healing and adaptive optimization. In *Proc. 25th Army Science Conference*.

MOLINO, N., BAO, Z., AND FEDKIW, R. 2004. A virtual node algorithm for changing mesh topology during simulation. *ACM Transactions on Graphics 23*, 3, 385–392.

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

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

NEALEN, A., MÜLLER, M., KEISER, R., BOXERMAN, E., AND CARLSON, M. 2006. Physically based deformable models in computer graphics. *Computer Graphics Forum 25*, 4, 809–836.

O'BRIEN, J. F., AND HODGINS, J. K. 1999. Graphical modeling and animation of brittle fracture. In *Proc. SIGGRAPH '99*, 137–146.

O'BRIEN, J. F., BARGTEIL, A. W., AND HODGINS, J. K. 2002. Graphical modeling and animation of ductile fracture. *ACM Transactions on Graphics 21*, 3, 291–294.

ODEN, J. T., AND DEMKOWICZ, L. F. 1989. Advances in adaptive improvements: A survey of adaptive finite element methods in computational mechanics. In *State-of-the-Art Surveys on Computational Mechanics*. The American Society of Mechanical Engineers, 441–467.

PARTHASARATHY, V. N., AND KODIYALAM, S. 1991. A constrained optimization approach to finite element mesh smoothing. *Finite Elements in Analysis and Design 9*, 4, 309–320.

PARTHASARATHY, V. N., GRAICHEN, C. M., AND HATHAWAY, A. F. 1994. A comparison of tetrahedron quality measures. *Finite Elements in Analysis and Design 15*, 3, 255–261.

SHAMIR, A., PASCUCCI, V., AND BAJAJ, C. 2000. Multiresolution dynamic meshes with arbitrary deformations. In *Proc. IEEE Visualization*, 423–430.

SHEWCHUK, J. R. 2002. Two discrete optimization algorithms for the topological improvement of tetrahedral meshes. Manuscript.

SIFAKIS, E., SHINAR, T., IRVING, G., AND FEDKIW, R. 2007. Hybrid simulation of deformable solids. In *Proc. Symposium on Computer Animation*, 81–90.

STEINEMANN, D., HARDERS, M., GROSS, M., AND SZEKELY, G. 2006. Hybrid cutting of deformable solids. In *Proc. IEEE Virtual Reality*.

STEINEMANN, D., OTADUY, M. A., AND GROSS, M. 2006. Fast arbitrary splitting of deforming objects. In *Proc. Symposium on Computer Animation*, 63–72.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. In *Proc. SIGGRAPH '87*, 205–214.

WOJTAN, C., AND TURK, G. 2008. Fast viscoelastic behavior with thin features. *ACM Transactions on Graphics 27*, 3, 47:1–47:8.

WOJTAN, C., THÜREY, N., GROSS, M., AND TURK, G. 2009. Deforming meshes that split and merge. *ACM Transactions on Graphics 28*, 3, 76:1–76:10.