Tetrahedral and Hexahedral Invertible Finite Elements

G. Irving *, J. Teran, R. Fedkiw

Stanford University

Abstract

We review an algorithm for the finite element simulation of elastoplastic solids which is capable of robustly and efficiently handling arbitrarily large deformation. In fact, the model remains valid even when large parts of the mesh are inverted. The algorithm is straightforward to implement and can be used with any material constitutive model, and for both volumetric solids and thin shells such as cloth. We also discuss a mechanism for controlling plastic deformation, which allows a deformable object to be guided towards a desired final shape without sacrificing realistic behavior, and an improved method for rigid body collision handling in the context of mixed explicit/implicit time-stepping. Finally, we present a novel extension of our method to arbitrary element types including specific details for hexahedral elements.

Key words:
PACS:

1 Introduction

Significant effort has been placed into making finite element simulation robust in the regime of large deformations, including the arbitrary Lagrangian-Eulerian (ALE) formulations pioneered by [1], continuous remeshing (see e.g. [2,3] and the references therein), etc. However, as noted in [4], these approaches are often computationally intensive and difficult to implement, which has primarily limited their use to two spatial dimensions. Moreover, [4] points out that these difficulties often lead authors to less optimal techniques such as

* Corresponding author.
Email addresses: irving@cs.stanford.edu (G. Irving), jteran@stanford.edu (J. Teran), fedkiw@cs.stanford.edu (R. Fedkiw).

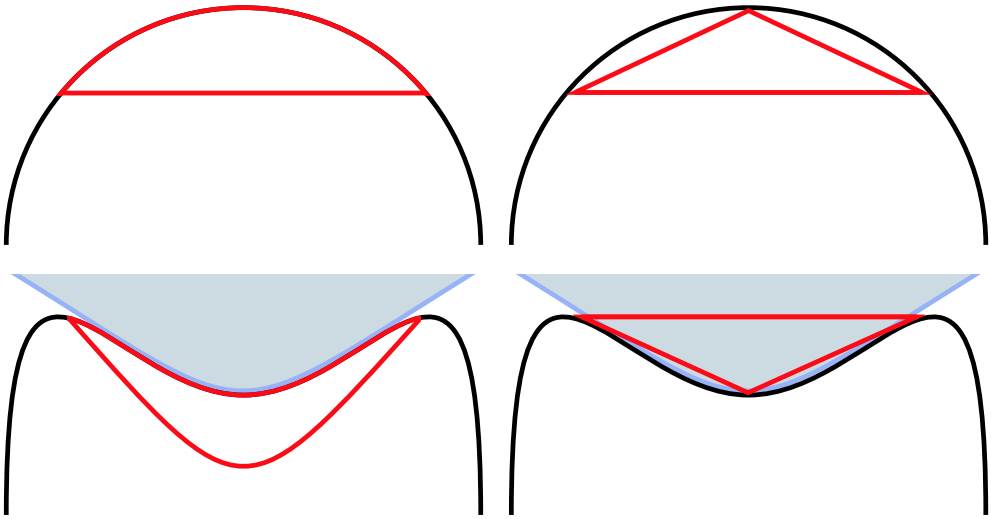

Fig. 1. A highly curved region of an object is pushed inwards during a collision with a grey object (left), and a triangle used to represent this deformation is forced to invert (right).

element deletion. This not only degrades the accuracy of the simulation, but is unsuitable for graphics applications where disappearing tetrahedra cause visual artifacts.

Since constitutive models for real materials are meaningful only for uninverted material, standard finite element simulation algorithms fail as soon as a single tetrahedron inverts. For this reason, various authors have proposed techniques for untangling inverted meshes. For example, [5] used feasible set methods and optimization to untangle two-dimensional meshes, [6] extended various tetrahedron quality metrics intended for mesh smoothing to the case of inverted tetrahedra allowing untangling to occur simultaneously with optimization, etc. However, none of these techniques are guaranteed to work, and fail quite often in practice especially if the boundary of the mesh is also tangled. And the failure to untangle a single tetrahedron forces the simulation to fail for most real world constitutive models.

As pointed out by [3], element inversion can occur even if the vertex positions of the mesh are identical to their true continuum values. A common case of this is illustrated in Figure 1 where a triangle with three nodes on the boundary is forced to invert during a collision with another object. Even if an object as a whole deforms by only a small amount, say 10%, an individual element may undergo severe deformation due to errors in the discrete representation of the continuous material. In fact, large deformation and inversion can arise even when simulating incompressible material, since one typically cannot conserve volume for each individual element. Given that it is difficult or impossible to prevent inversion in all cases, we propose a simpler approach that allows elements to invert gracefully and recover.

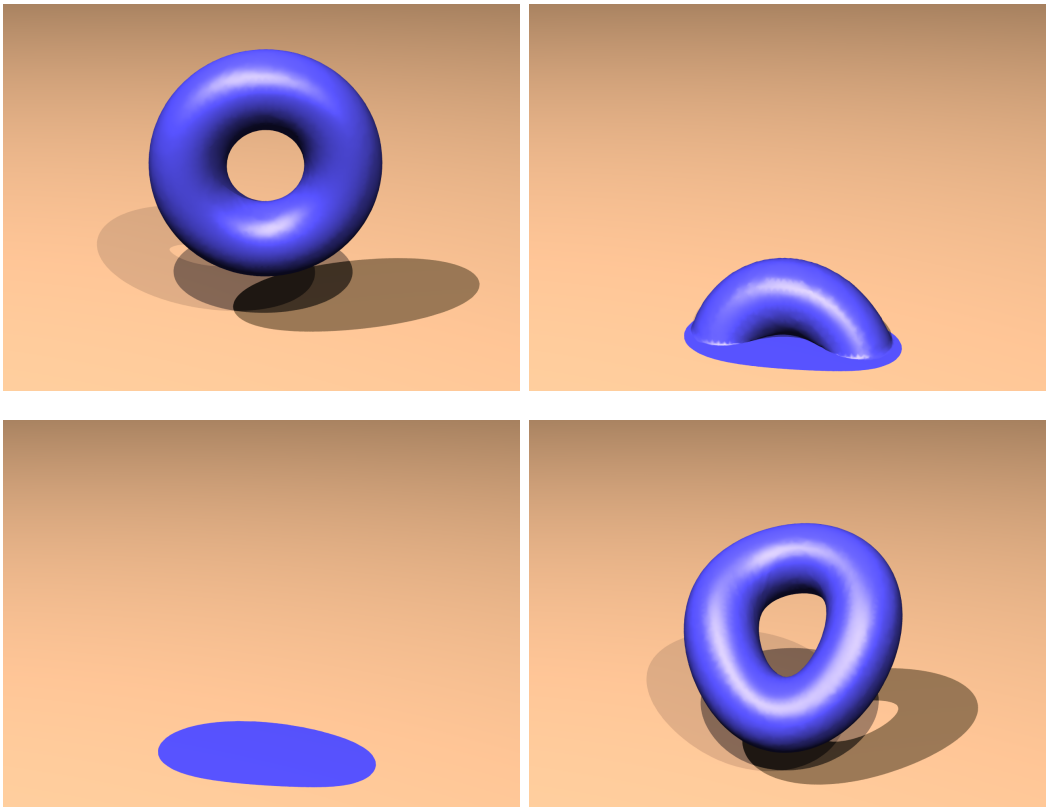

Fig. 2. A torus with zero strength collapses into a puddle. When the strength is increased, the torus recovers.

If the particular material behavior and underlying physics is unimportant, there are several techniques for treating inversion. For mass-spring systems, altitude springs work surprisingly well [7] (see also the work on "Van Helsing" [8]). In fact, one can even add altitude springs to finite element models, but this changes the underlying partial differential equations and thus the behavior of the material (losing one of the key benefits of the finite element method as compared to mass-spring systems). Moreover, one cannot use constitutive models that lose meaning for inverting elements (which is most of them). Of course, one could switch from finite element methods to mass-spring methods for flat, degenerate, and inverted elements but this leads to force discontinuities (detrimental to implicit time integration or quasistatic simulation), visual artifacts such as popping, etc. Various authors have proposed methods similar in spirit to altitude springs. [9] changes the underlying partial differential equation by adding a volume preservation term to penalize inversion. [10] first computes and removes the rotation from material to spatial coordinates, and then applies a linear model in the unrotated space. Although the linear model works well for inverted elements, it is severely limited in the types of materials it can approximate. In fact, we stress that a spring is the most common linear model that correctly accounts for rotation.

In [11], we took a different approach motivated by the desire to continuously

(or smoothly) extend the finite element method so that it behaves gracefully for both degenerate and inverted elements, even for arbitrary constitutive models. This is especially important as a number of recent graphics publications have advocated the use of more advanced constitutive models to capture more realistic physical behavior. Moreover, many materials found in nature exhibit complex nonlinearities under large deformation, such as the biphasic nature of biological tissue and the anisotropic behavior of muscle. Our approach begins by computing a diagonalization of the deformation mapping in order to determine the "direction" along which a given tetrahedron is inverted. The constitutive model is then extended into the inverted regime using C^0, C^1 or higher continuity around the flat state, resulting in smooth behavior even in extreme situations. The resulting forces *always* act to restore the tetrahedron to its original shape, allowing objects to recover cleanly from flat or inverted configurations, as shown in Figure 2. The generality and robustness of the approach was illustrated with a number of simulations of objects undergoing large deformations, including nonlinear and anisotropic constitutive models, plasticity with and without control, both volumetric objects and thin shells, and fracture.

This paper is an extended version of [11] including a review of the material presented therein. In addition, we also present an extension to arbitrary element types with specific details and simulation results for hexahedral elements. This is readily accomplished because the modifications required to handle inversion involve only the first Piola-Kirchoff stress $\mathbf{P}$ which is independent of the element type. Therefore, an arbitrary element can be made robust through inversion by modifying the computation of $\mathbf{P}$ at each quadrature point.

2 Related Work

[12–14] pioneered deformable models in computer graphics including early work on plasticity and fracture. Finite element simulations have been used to model a hand grasping a ball [15], to simulate muscles [16], for virtual surgery [17], and to simulate data from the NIH visible human data set [18,19]. [20] and [21] simulated brittle and ductile fracture respectively, while [22] coupled this work to explosions. Finite elements were also used for fracture in [23]. Other work includes the adaptive framework of [24], the rotation based approach in [25] and [10], the hybrid finite element free form deformation approaches in [26,27], and the finite volume muscle models of [28]. Other interesting approaches to the simulation of deformable objects include [29,30].

Many authors have worked to improve the robustness of mass-spring systems. [31] used a pseudopressure term in addition to edge springs, [32] used springs emanating from the barycenter of each tetrahedron to preserve volume, and

[33] introduced altitude springs to prevent triangles from collapsing. [7] later improved this model and applied it in three spatial dimensions to the case of tetrahedral mesh generation. If altitude springs are used correctly, not only is inversion not a problem, but the elements will work to un-invert. Unfortunately, spring systems do not allow the modeling of arbitrary constitutive models.

We also show examples of our method at work for the in-plane deformations of lower dimensional manifolds such as cloth and shells. Here, since triangles cannot invert in three spatial dimensions, our method is similar to the work of [34], except that they do not consider degenerate elements. For out-of-plane forces, we use the bending model of [35] (see also [36]), and for self-collisions we use the method of [37]. Moreover, for volumetric collisions we also use the method in [37] simply applied to the triangulated boundary surface of the tetrahedron mesh. Other interesting work on cloth and shells includes the implicit time stepping of [38], the bending model of [39], the adaptive simulation work of [40], and the self-collision untangling strategy of [41].

3 Measuring deformation

A deformable object is characterized by a time dependent map ϕ from material coordinates $\mathbf{X}$ to world coordinates $\mathbf{x}$. The stress at a given point $\mathbf{X}$ in the material depends only on the deformation gradient $\mathbf{F}(\mathbf{X}) = \partial\mathbf{x}/\partial\mathbf{X}$ of this mapping. Since we are using a purely Lagrangian framework, all mappings are based in material space. In order to discretely represent ϕ, material space is divided into finite elements such as tetrahedrons or hexahedrons. In order to interpolate values defined on vertices in a consistent manner, we make use of isoparametric elements parameterized by ξ [42]. The point is that nodal values of a variable $\mathbf{x}_i$ can be expressed throughout the element via

$$\mathbf{x}(\xi) = \sum_{i=1}^{n_e} \mathbf{x}_i N_i(\xi)$$

where n_e is the number of vertices in the element and each $N_i(\xi)$ is an interpolating function associated with node i. Using this and the chain rule allows us to compute the deformation gradient as

$$\mathbf{F} = \frac{\partial\mathbf{x}}{\partial\mathbf{X}} = \frac{\partial\mathbf{x}}{\partial\xi}\left(\frac{\partial\mathbf{X}}{\partial\xi}\right)^{-1} = \sum_{i=1}^{n_e} \mathbf{x}_i \frac{\partial N_i(\xi)}{\partial\xi}\left(\sum_{i=1}^{n_e} \mathbf{X}_i \frac{\partial N_i(\xi)}{\partial\xi}\right)^{-1}.$$

For simplicity we assemble the spatial positions of the element vertices in a $3 \times n_e$ matrix $\mathbf{D}_s = [\mathbf{x}_1, ..., \mathbf{x}_{n_e}]$, and similarly for material positions, $\mathbf{D}_m = [\mathbf{X}_1, ..., \mathbf{X}_{n_e}]$. Additionally, we assemble the derivatives $\frac{\partial N_i}{\partial\xi}$ in a $n_e \times 3$ matrix

$\mathbf{H} = [\frac{\partial N_1}{\partial \xi}^T, \cdots, \frac{\partial N_{n_e}}{\partial \xi}^T]^T$. With these conventions, $\mathbf{F}$ can be written as $\mathbf{F} = \mathbf{D}_s\mathbf{H}(\xi)\left(\mathbf{D}_m\mathbf{H}(\xi)\right)^{-1}$.

In order to estimate nodal forces for a given element, we must evaluate $\mathbf{F}$ at several quadrature points ξ_g. Since the element is fixed in material coordinates, $\mathbf{H}(\xi_g)\left(\mathbf{D}_m\mathbf{H}(\xi_g)\right)^{-1}$ is constant and can be precomputed for efficiency. More importantly, as long as the *initial* mesh is reasonable, the $n_e \times 3$ matrix $\mathbf{H}(\xi_g)\left(\mathbf{D}_m\mathbf{H}(\xi_g)\right)^{-1}$ is well-conditioned, and therefore $\mathbf{F}_g = \mathbf{D}_s\mathbf{H}(\xi_g)\left(\mathbf{D}_m\mathbf{H}(\xi_g)\right)^{-1}$ is well-defined and finite regardless of the current state of the object. Furthermore, the values of $\mathbf{F}_g$ at the quadrature points contain all the information about the deformation of each element. In particular, we can monitor inversion at each quadrature point simply by checking the sign of $\det \mathbf{F}_g$.

Once we have $\mathbf{F}$, the next step is usually to define the Green strain $\mathbf{G} = 1/2(\mathbf{F}^T\mathbf{F} - \mathbf{I})$, and compute stress and forces based on $\mathbf{G}$. We do not do this, however, since $\mathbf{G}$ is invariant with respect to all orthogonal transformations, including reflection, and is therefore incapable of detecting inversion. Furthermore, $\mathbf{G}$ is already nonlinear in the deformation, and it is therefore more difficult to interpret the large deformation behavior of a constitutive model based on $\mathbf{G}$ than one based on $\mathbf{F}$, which is linearly related to deformation. Thus, for the remainder of this paper, we make the (nonrestrictive) assumption that the constitutive model is written explicitly in terms of $\mathbf{F}$.

In the case of tetrahedral elements, we use barycentric interpolation resulting in constant deformation gradients, and only one quadrature point is required. As a result the 4×4 matrix $\mathbf{H}$ has a particularly simple structure consisting of the identity matrix atop a row consisting of all -1 entries. Therefore, in [11], we absorbed $\mathbf{H}$ into $\mathbf{D}_s$ and $\mathbf{D}_m$ to obtain $\mathbf{D}_s = [\mathbf{x}_2 - \mathbf{x}_1, \mathbf{x}_3 - \mathbf{x}_1, \mathbf{x}_4 - \mathbf{x}_1]$, $\mathbf{D}_m = [\mathbf{X}_2 - \mathbf{X}_1, \mathbf{X}_3 - \mathbf{X}_1, \mathbf{X}_4 - \mathbf{X}_1]$ and $\mathbf{F} = \mathbf{D}_s\mathbf{D}_m^{-1}$. Furthermore, if the material is isotropic, we can save storage space by performing a QR-decomposition of $\mathbf{D}_m$ and storing only the upper triangular part, as noted in [23]. This corresponds to rotating material space, and therefore has no effect on an isotropic material. This optimization can be performed for an anisotropic model by rotating the anisotropic terms via the rotation from the QR-decomposition.

Hexahedral elements are also commonly used and are determined by trilinearly interpolating nodal values throughout the primitive element $[-1, 1]^3$. The associated interpolating functions are given as

$$N_{4(i-1)+2(j-1)+k}(\xi) = \frac{(1 + (-1)^i\xi_1)(1 + (-1)^j\xi_2)(1 + (-1)^k\xi_3)}{8}$$

where $i, j, k = 1, 2$. Note that these elements do not yield a compact expression for the deformation gradient as in the case of tetrahedral elements. Additionally, the deformation gradient typically needs to be evaluated at 8

different quadrature points within each element making hexahedral elements considerably more expensive than tetrahedral elements.

4 Force computation

When the constitutive model is given as a first Piola-Kirchhoff stress $\mathbf{P}$, an element's contribution to the finite element force on one of its nodes $\mathbf{x}_a$ is given as

$$\mathbf{f}_a^e = \int_{\Omega_m^e} \mathbf{P}\frac{\partial N_a}{\partial \mathbf{X}}^T d\mathbf{X} = \int_{\Omega_i} \mathbf{P}\frac{\partial N_a}{\partial \mathbf{X}}^T \left|\frac{\partial \mathbf{X}}{\partial \xi}\right| d\xi$$

where Ω_m^e represents the element in material space and Ω_i represents the ideal element. This integral can be approximated using quadrature as

$$\mathbf{f}_a^e = \sum_{g=1}^{n_g} \mathbf{P}_g \left(\frac{\partial N_a}{\partial \mathbf{X}}\right)_g^T \left|\left(\frac{\partial \mathbf{X}}{\partial \xi}\right)_g\right| W_g = \sum_{g=1}^{n_g} \mathbf{P}_g (\mathbf{D}_m \mathbf{H}_g)^{-T} \left(\frac{\partial N_a}{\partial \xi}\right)_g^T \left|\left(\frac{\partial \mathbf{X}}{\partial \xi}\right)_g\right| W_g$$

where where n_g is the number of quadrature points, $\mathbf{P}_g$ is the first Piola-Kirchhoff stress at a quadrature point g, and W_g is the weight associated with quadrature point g. The second equality comes from $\frac{\partial N_a}{\partial \mathbf{X}} = \frac{\partial N_a}{\partial \xi}\left(\frac{\partial \mathbf{X}}{\partial \xi}\right)^{-1} = \frac{\partial N_a}{\partial \xi}(\mathbf{D}_m \mathbf{H})^{-1}$. The element's contribution to *all* its nodal forces can be compactly represented as $\mathbf{G} = [\mathbf{f}_1^e, \cdots, \mathbf{f}_{n_e}^e] = \sum_{g=1}^{n_g} \mathbf{G}_g$ where

$$\mathbf{G}_g = \mathbf{P}_g (\mathbf{D}_m \mathbf{H}_g)^{-T} \mathbf{H}_g^T \left|\left(\frac{\partial \mathbf{X}}{\partial \xi}\right)_g\right| W_g = \mathbf{P}_g \mathbf{B}_{mg}$$

using the definition of $\mathbf{H}$. Note that since the $3 \times n_e$ matrix $\mathbf{B}_{mg}$ is constant, the nodal forces are linearly related to $\mathbf{P}$. Therefore, the key to obtaining robust forces in the face of large deformation is an accurate calculation of $\mathbf{P}$.

If a constitutive model is given in terms of a Cauchy stress σ or second Piola-Kirchhoff stress $\mathbf{S}$, we can easily convert to a first Piola-Kirchhoff stress via the formulas $\mathbf{P} = \mathbf{FS}$ and $\mathbf{P} = J\sigma\mathbf{F}^{-T}$ where $J = \det \mathbf{F}$. Alternatively, one could rewrite the force formula $\mathbf{G}_g = \mathbf{P}_g \mathbf{B}_{mg}$ directly in terms of the other stresses as $\mathbf{G}_g = \sigma_g \mathbf{B}_{sg}$ or $\mathbf{G}_g = \mathbf{F}_g \mathbf{S}_g \mathbf{B}_{mg}$. Unlike the first Piola-Kirchhoff case where obtaining a valid $\mathbf{P}_g$ is sufficient to obtain robust forces, computing a valid σ_g or $\mathbf{S}_g$ is not enough. For example, if the element is a single point, $\mathbf{F}_g$ and thus $\mathbf{G}_g = \mathbf{F}_g \mathbf{S}_g \mathbf{B}_{mg}$ are identically zero. In such instances there are no restorative forces. Therefore, we write all constitutive models in terms of $\mathbf{P}$ before force computation.

In the case of tetrahedral elements, only one quadrature point is needed per element and an element's contribution to the elastic force on node i is equiva-

lent to $\mathbf{g}_i = -\mathbf{P}\left(A_1\mathbf{N}_1 + A_2\mathbf{N}_2 + A_3\mathbf{N}_3\right)/3$, where $A_j\mathbf{N}_j$ are the area weighted normals (in material coordinates) of the faces of the tetrahedron incident to node i. This was shown in [28] and results from the fact that a first Piola-Kirchhoff stress $\mathbf{P}$ is a mapping from area-weighted normals in material space to traction vectors in world space. See also [11] for more details.

For hexahedral elements, 8 quadrature points per element are typically used. Particularly common is the second order quadrature that uses the points

$$\xi_{4(i-1)+2(j-1)+k} = \left[(-1)^i\frac{1}{\sqrt{3}}, (-1)^j\frac{1}{\sqrt{3}}, (-1)^k\frac{1}{\sqrt{3}}\right]$$

in the ideal domain with weights $W_{4(i-1)+2(j-1)+k} = 1$ where $i, j, k = 1, 2$.

5 Diagonalization

Since rigid body rotations do not change the physics of a deformable object, the stress $\mathbf{P}$ satisfies $\mathbf{P}(\mathbf{UF}) = \mathbf{UP}(\mathbf{F})$ for any rotation $\mathbf{U}$. (Here $\mathbf{P}(\mathbf{F})$ denotes function application.) Furthermore, if we temporarily assume an isotropic constitutive model, $\mathbf{P}$ is invariant under rotations of material space, i.e. $\mathbf{P}(\mathbf{FV}^T) = \mathbf{P}(\mathbf{F})\mathbf{V}^T$. Therefore, if we diagonalize $\mathbf{F}$ via rotations $\mathbf{U}$ and $\mathbf{V}$ to obtain $\mathbf{F} = \mathbf{U}\hat{\mathbf{F}}\mathbf{V}^T$, $\mathbf{P}$ becomes

$$\mathbf{P} = \mathbf{P}(\mathbf{F}) = \mathbf{UP}(\hat{\mathbf{F}})\mathbf{V}^T = \mathbf{U}\hat{\mathbf{P}}\mathbf{V}^T \tag{1}$$

where a hat superscript denotes the corresponding rotated quantity. Since the elastic energy of an isotropic material is invariant under world and material rotations, it can depend only on the invariants of $\mathbf{F}$, or equivalently on the entries of the diagonalization $\hat{\mathbf{F}}$ (see e.g. [43]). Therefore, the gradient of the energy, $\hat{\sigma}$, will also be diagonal. Moreover, since the three stresses are related via $\hat{\sigma} = (1/J)\hat{\mathbf{P}}\hat{\mathbf{F}}^T$ and $\hat{\mathbf{P}} = \hat{\mathbf{F}}\hat{\mathbf{S}}$, the diagonalization of $\mathbf{F}$ actually results in the simultaneous diagonalization of all three stresses. In particular, $\hat{\mathbf{P}}$ in equation 1 is diagonal for an isotropic constitutive model. For an anisotropic constitutive model, a diagonal $\hat{\mathbf{F}}$ does not result in a diagonal $\hat{\mathbf{P}}$. However, this is not restrictive, and we show examples of anisotropic constitutive models in section 6.1.

The diagonalization of $\mathbf{F}$ is not unique, however. While the ordering of the entries of the diagonal matrix $\hat{\mathbf{F}}$ is unimportant, the signs of the entries determine the particular direction of inversion. The standard SVD convention of choosing all nonnegative entries works only when $\det\mathbf{F} \geq 0$. When $\det\mathbf{F} < 0$, the signs of the entries must be chosen carefully in order to guarantee that the forces act to uninvert the element. In this case, $\hat{\mathbf{F}}$ has either one or three negative entries. We heuristically assume that each element is as uninverted

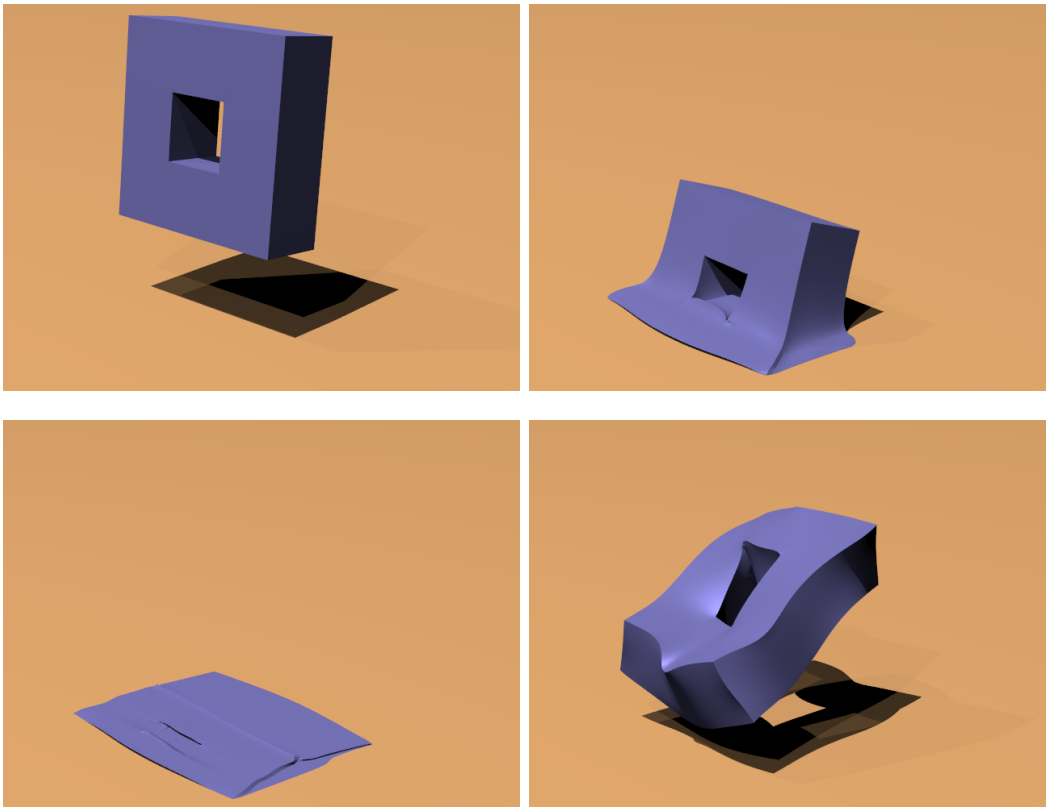

Fig. 3. A hexahedron mesh collapses into a puddle and recovers.

as possible, and thus we assume that only one entry (not three) is negative. Moreover, the entry with the smallest magnitude is chosen to be negative. This is motivated by the geometric fact that an inverted tetrahedron can be uninverted by moving any one node across the plane of the opposite face, and it is most efficient to choose the node that is closest to the opposite face.

We compute the correct diagonalization by finding any diagonalization and correcting the signs. When doing this, we must be careful to ensure that the final $\mathbf{U}$ and $\mathbf{V}$ are pure rotations, i.e., $\det \mathbf{U} = \det \mathbf{V} = 1$. This is because deformable objects are not invariant under reflections of material or world space, and equation 1 does not hold if either $\mathbf{U}$ or $\mathbf{V}$ is a reflection. We compute the SVD of $\mathbf{F} = \mathbf{U}\hat{\mathbf{F}}\mathbf{V}^T$ as follows. First we form the normal equations $\mathbf{F}^T\mathbf{F} = \mathbf{V}\hat{\mathbf{F}}\mathbf{U}^T\mathbf{U}\hat{\mathbf{F}}\mathbf{V}^T = \mathbf{V}\hat{\mathbf{F}}^2\mathbf{V}^T$. Then we rearrange to obtain an eigenproblem, $\mathbf{F}^T\mathbf{F}\mathbf{V} = \mathbf{V}\hat{\mathbf{F}}^2$ for the symmetric positive semidefinite $\mathbf{F}^T\mathbf{F}$. Here, $\mathbf{V}$ is an orthogonal matrix of eigenvectors and $\hat{\mathbf{F}}^2$ is a diagonal matrix with nonnegative entries. Robust computation of eigensystems for 3×3 matrices (even with repeated or zero eigenvalues) is a solved problem. And since it is relevant to rigid body simulations, it has received a lot of attention. Note that if $\mathbf{V}$ is a reflection with $\det \mathbf{V} = -1$ we can simply multiply a column of $\mathbf{V}$ by -1 to make $\mathbf{V}$ a rotation with $\det \mathbf{V} = 1$. The entries of $\hat{\mathbf{F}}$ are then determined by taking the square root of the diagonal elements of $\hat{\mathbf{F}}^2$, and $\mathbf{U}$ can be found via $\mathbf{U} = \mathbf{F}\mathbf{V}\hat{\mathbf{F}}^{-1}$ for well shaped elements. However, if a diagonal entry of $\hat{\mathbf{F}}$

is near zero, we do not use this formula for the corresponding column of $\mathbf{U}$, but instead take it to be orthogonal to the other columns. For example, in the extreme case where $\mathbf{F} = \mathbf{0}$, we choose $\mathbf{U} = \mathbf{I}$. Finally, to treat inversion where $\det \mathbf{F} < 0$, we have $\det \mathbf{U} = -1$ implying that $\mathbf{U}$ is a reflection. This is removed by negating the minimal element of $\hat{\mathbf{F}}$ and the corresponding column of $\mathbf{U}$. Figure 2 illustrates degeneracy and inversion handling for a tetrahedral torus mesh. Moreover, we have tested our approach for a variety of degenerate configurations, such as when a tetrahedron collapses to a single point or line, and the method always leads to robust recovery from inversion. Figure 3 shows similar results for a hexahedral mesh.

5.1 Other Rotations

Typically, authors use a polar decomposition to remove the world rotation of a tetrahedron producing a symmetric $\mathbf{F}_s$ with $\mathbf{F} = \mathbf{Q}\mathbf{F}_s$. To recover from inversion, one must be careful to control the signs of the eigenvalues of $\mathbf{F}_s$ as in the diagonalization case. However, we know of no way to do this without first computing the full diagonalization $\mathbf{F} = \mathbf{U}\hat{\mathbf{F}}\mathbf{V}^T$, and forming the polar decomposition via $\mathbf{Q} = \mathbf{U}\mathbf{V}^T$, $\mathbf{F}_s = \mathbf{V}\hat{\mathbf{F}}\mathbf{V}^T$. Polar decomposition was used by [34] for cloth simulation and [10] for volumetric solids, but neither showed how to correctly handle inverting or degenerate elements. However, for the cloth case, we note that triangles do not invert in three spatial dimensions (although they can become degenerate). Both tetrahedra in three spatial dimensions and triangles in two spatial dimensions can invert and become degenerate.

Alternatively, one could attempt to remove the world rotation with a QR-decomposition, i.e. $\mathbf{F} = \mathbf{Q}\mathbf{F}_r$ with $\mathbf{F}_r$ an upper triangular matrix. However, any stress which depends linearly on $\mathbf{F}_r$ will be anisotropic in a mesh dependent way, since it is not invariant under rotations of material space. To see this, it suffices to note that if $\mathbf{V}$ is a rotation of material space, then $\mathbf{F}\mathbf{V}^T = \mathbf{Q}\mathbf{F}_r\mathbf{V}^T$, and $\mathbf{F}_r\mathbf{V}^T$ is not upper triangular. Therefore, QR-decomposition is inadvisable even if physical accuracy is not a requirement. Moreover, this is a problem with any method for removing rotations, such as [25], that does not use $\mathbf{Q} = \mathbf{U}\mathbf{V}^T$ for the world rotation.

Note that our approach departs from the typical goal of determining (or approximating) the rotation from material space to world space, i.e. $\mathbf{Q}$ from the polar decomposition. Instead, we look for *two* rotations $\mathbf{U}$ and $\mathbf{V}$ such that $\mathbf{U}^T$ and $\mathbf{V}^T$ rotate the world and material spaces, respectively, to a space where the deformation gradient is a diagonal matrix. This is preferable to the space obtained using $\mathbf{Q}$ in which the deformation gradient is a more complex symmetric matrix.

6 Constitutive Models

Once we have carefully diagonalized $\mathbf{F}$, we can extend our constitutive models to behave reasonably under inversion. The diagonal setting makes this quite simple.

If St. Venant-Kirchhoff material is compressed beyond a certain point, it gets weaker and weaker as the compression increases, and the stress drops to zero as the object becomes flat. Moreover, if an element inverts, the forces act to keep the element inverted. See Figure 4 (upper left). As noted in [43], this makes the St. Venant-Kirchhoff model completely useless for modeling large deformations. [20] noted these difficulties, but dismissed them since they were simulating rigid materials. However, as discussed previously, stiff or incompressible objects may still have inverted elements due to discretization error, especially on the coarse grids common in the computer graphics community.

In order to alleviate the problems with the St. Venant-Kirchhoff model, various authors (e.g. [17]) have proposed adding a pseudo-pressure term to prevent element inversion. In fact, the classical neo-Hookean constitutive model already does this as shown in Figure 4 (upper right). The singularity at the origin means that infinite energy is required to completely flatten an element, and as long as the equations for this constitutive model are accurately simulated, inversion is prevented. However, preventing inversion also prevents the handling of situations where inversion is the desired, correct response, as in Figure 1. Moreover, since the forces become arbitrarily large, the system can become arbitrarily stiff and difficult to integrate, making it difficult to handle situations such as that shown in Figure 7 where a volumetric Buddha model is pulled through rigid, interlocking gears.

To avoid the unnecessary stiffness associated with the neo-Hookean constitutive model, we modify the constitutive model near the origin to remove the singularity by either linearizing at a given compression limit or simply clamping the stress at some maximum value. Moreover, as shown in Figure 4 (lower right), we extend the model past the origin into the inverted regime in order to obtain valid forces for inverted elements. These forces act to uninvert the element. Note that since we have removed both spatial and material rotations by diagonalizing, the modified model is automatically rotation invariant and isotropic.

The major strength of the diagonal setting is that these modifications can be applied to arbitrary constitutive models. This is quite natural, since the diagonal setting is also commonly used in the experimental determination of material parameters. The resulting model is identical to the physical model most of the time, and allows the simulation to continue if a few elements

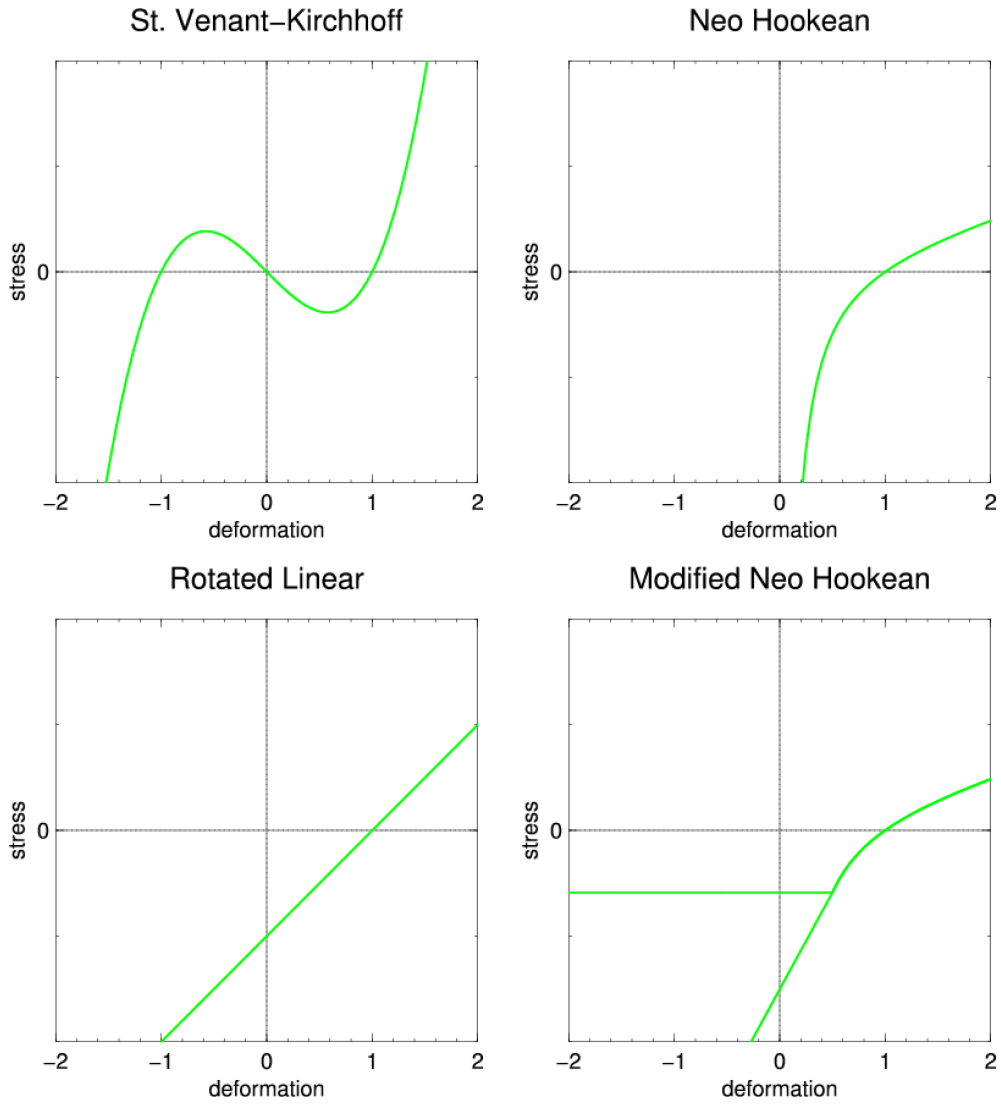

Fig. 4. The relationship between the first Piola-Kirchhoff stress $\hat{\mathbf{P}}$ and the deformation gradient $\hat{\mathbf{F}}$ for various constitutive models.

invert. Furthermore, our extensions provide C^0 or C^1 continuity around the flat case, which avoids sudden jumps or oscillations which might effect neighboring elements.

While it may seem nonphysical to modify a constitutive model for inversion handling, most constitutive models lose accuracy long before inversion occurs. It is exceedingly difficult to measure material response in situations of extreme compression, so constitutive models are often measured for moderate deformation and continued heuristically down to the flat cases. Given that some accuracy loss is unavoidable when elements are extremely degenerate, it is preferable to provide smooth, consistent handling of inversion in order to avoid unnecessary corruption of the more meaningful parts of the simulation.

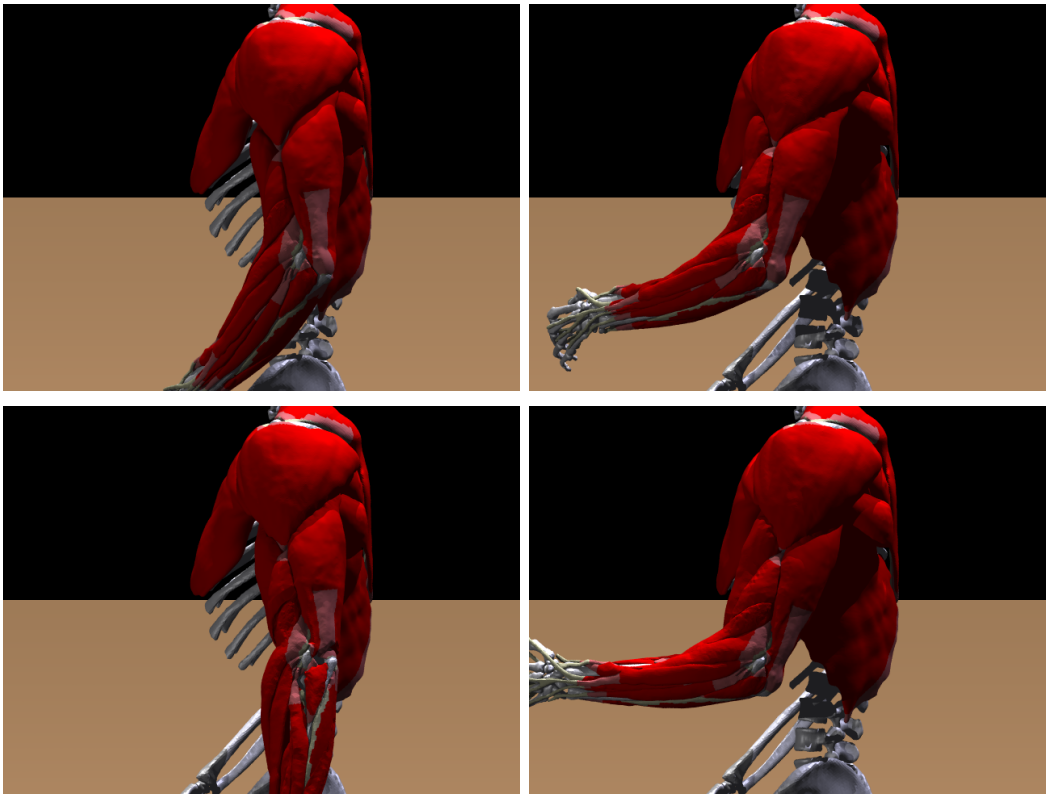

Fig. 5. A simulation of muscles driven by a key-framed skeleton. The muscle is represented with a transversely isotropic constitutive model, and the strength along the fiber direction in the muscle is based on activation levels.

If a specific qualitative material behavior is desired but the exact quantitative model is less important, we can use the diagonal setting to construct a suitable constitutive model. For example, most biological material is soft under small deformation, but becomes stiffer as the deformation increases. A simple model capturing this behavior is given by choosing threshold values for compression and elongation, specifying the slope of the stress curve outside these threshold values and at the undeformed state, and using a cubic spline to interpolate between them. This model, equipped with a linear pressure component, was used for the simulations of the volumetric Buddha model shown in Figures 6 and 7, and the hexahedral simulation in Figure 8. Note that any isotropic constitutive model expressed in diagonal form will automatically preserve angular momentum, since if $\hat{\mathbf{P}}$ is diagonal, $\hat{\mathbf{S}} = \hat{\mathbf{F}}^{-1}\hat{\mathbf{P}}$ is symmetric (see e.g. [43]). For the torus puddle, hexahedron puddle, and plastic sphere simulations (see Figures 2 and 9), where the focus is on degeneracy and plasticity, respectively, we used the simple rotated linear model $\hat{\mathbf{P}} = 2\mu(\hat{\mathbf{F}} - \mathbf{I}) + \lambda tr(\hat{\mathbf{F}} - \mathbf{I})$ depicted in Figure 4 (lower left).

Once we have computed the diagonalized stress $\hat{\mathbf{P}}$ at a given quadrature point, the force computation becomes

$$\mathbf{G} = \mathbf{U}\hat{\mathbf{P}}\mathbf{V}^T\mathbf{B}_m = \mathbf{U}\hat{\mathbf{P}}\hat{\mathbf{B}}_m \tag{2}$$

where $\hat{\mathbf{B}}_m = \mathbf{V}^T\mathbf{B}_m$ can be computed and stored if the rotation is fixed for multiple force computations, as in some versions of Newmark time integration (see section 9).

6.1 Anisotropy

If the constitutive model includes anisotropic components, it is no longer invariant under rotations of material space. However, we can continue to fully diagonalize $\mathbf{F}$, and rotate the anisotropic terms using $\mathbf{V}$. Since we still work with a diagonal $\hat{\mathbf{F}}$, the large deformation behavior of the constitutive model is still apparent and easy to modify to handle inversion. For example, if the material is stronger in a certain material direction $\mathbf{a}$, we diagonalize $\mathbf{F}$ and use $\mathbf{V}^T\mathbf{a}$ in the computation of $\hat{\mathbf{P}}$. $\hat{\mathbf{P}}$ is no longer a diagonal matrix, but we can still compute forces using equation 2. When constructing anisotropic constitutive models that allow inversion, we write $\hat{\mathbf{P}}$ as a diagonal matrix plus $\hat{\mathbf{F}}$ times a symmetric matrix for the anisotropic terms. Then $\hat{\mathbf{S}} = \hat{\mathbf{F}}^{-1}\hat{\mathbf{P}}$ is symmetric (preserving angular momentum) as required.

We illustrate the handling of anisotropy with an example simulation of skeletal muscle in the upper limb (see Figure 5). We use a nonlinear transversely-isotropic quasi-incompressible constitutive model. See [44] for more details. This is an intricate region of the body articulated with complex joints in the shoulder, elbow and wrist. Inaccuracy in the joint models and motion data leads to skeletal configurations that are incompatible with the musculature creating boundary conditions that degenerately deform muscles and tendons leading to spurious element inversion. However, these configurations often only occur in limited regions of the mesh and only for brief moments during a given motion. Our algorithm allows simulations to progress past these temporary problems by letting elements invert and then later recover.

6.2 Damping

Damping forces can be implemented by rotating the velocity gradient $\dot{\mathbf{F}}$ by the same $\mathbf{U}$ and $\mathbf{V}$ used to diagonalize $\mathbf{F}$, computing the damping stress $\hat{\mathbf{P}}$ in the rotated frame, and computing the force exactly as for the elastic case. Note that the rotated velocity gradient will in general not be diagonal.

As in the case of anisotropic elastic forces, a damping model will only preserve angular momentum if $\hat{\mathbf{P}}$ can be expressed as $\hat{\mathbf{F}}\hat{\mathbf{S}}$, with $\hat{\mathbf{S}}$ symmetric. This was not a problem for anisotropy since the anisotropic terms are usually not important for flat or inverted elements. However, in order to prevent visually unpleasant oscillations, we do not want the damping forces to disappear for

flat elements. For example, the analogous damping model to the rotated linear constitutive model, $\hat{\mathbf{P}} = \beta(\hat{\dot{\mathbf{F}}} + \hat{\dot{\mathbf{F}}}^T) + \alpha tr(\hat{\dot{\mathbf{F}}})$, does not preserve angular momentum unless $\hat{\mathbf{F}}$ is a uniform scaling. However, since the angular momentum errors are small around the undeformed state, and highly deformed elements are usually interacting with other objects, we have not found this lack of conservation to be visually noticeable. In simulations where more physical accuracy is desired, we use a correct damping model for moderate deformations and a more robust but nonphysical model for the few flat or inverted elements.

6.3 Plasticity

We represent plastic deformation with a multiplicative decomposition of the deformation $\mathbf{F} = \mathbf{F}_e\mathbf{F}_p$, where $\mathbf{F}_p$ represents the permanent plastic deformation and $\mathbf{F}_e$ the elastic deformation, see e.g. [43] or [45]. The multiplicative formulation allows a complete separation between plastic flow and elastic forces, and makes constraints such as volume preservation simple to enforce. In contrast, the additive plasticity formulation of [21] does not support true incompressibility, though this might not be a significant problem for graphics applications. Note that if the elastic constitutive model is isotropic, the rotational part of $\mathbf{F}_p$ is arbitrary, e.g. we can choose $\mathbf{F}_p$ to be symmetric.

We restrict ourselves to rate-independent plasticity models, and use the return mapping algorithm to transfer deformation from the elastic part $\mathbf{F}_e$ to the plastic part $\mathbf{F}_p$ whenever a yield criterion on $\mathbf{F}_e$ is exceeded. The details of the computation of plastic flow are as follows. Compute the trial elastic deformation $\mathbf{F}_{e,trial} = \mathbf{F}\mathbf{F}_p^{-1}$, and find the diagonalization $\mathbf{F}_{e,trial} = \mathbf{U}\hat{\mathbf{F}}_{e,trial}\mathbf{V}^T$. If a yield criterion on $\hat{\mathbf{F}}_{e,trial}$ is exceeded, project back onto the yield surface producing a new diagonal matrix $\hat{\mathbf{F}}_{e,proj}$. Compute the trial plastic deformation $\mathbf{F}_{p,trial} = \hat{\mathbf{F}}_{e,proj}^{-1}\mathbf{U}^T\mathbf{F}$ (dropping $\mathbf{V}$ since rotations of $\mathbf{F}_p$ are unimportant). If $\mathbf{F}_{p,trial}$ exceeds a separate limit criterion, project it back onto the limit surface producing the final $\mathbf{F}_p$. Compute and store $\mathbf{F}_p^{-1}$ for future use.

This structure supports an arbitrary plastic yield criterion while still ensuring that the plastic deformation does not become too extreme. This is important, since the time step required for stability depends on the conditioning of $\mathbf{F}_p$. In particular, $\mathbf{F}_p$ should never invert. This can be implemented in the diagonal framework by ensuring that the projection of $\hat{\mathbf{F}}_e$ always results in an $\hat{\mathbf{F}}_{e,proj}$ with positive entries. The final limiting of $\mathbf{F}_{p,trial}$ can then be adjusted to ensure a well-conditioned $\mathbf{F}_p$. Figure 11 shows a simulation of ductile fracture using this technique. See [46] for more details.

7 Controlling Plasticity

Various authors, such as [47–49], have considered controlling physics based simulations. The ability to control a simulation alleviates the need for laborious parameter tuning to achieve a desired effect, and makes possible animations which could not be achieved through physical accuracy alone. In the context of plasticity, we can use the plastic limiting projection step to control the plastic deformation toward any desired state without sacrificing realism. To do this, we compute a goal deformation $\mathbf{F}_{p,goal}$ at the beginning of the simulation. In the plastic projection step, we are given a tentative plastic flow from the old deformation $\mathbf{F}_p$ to the trial deformation $\mathbf{F}_{p,trial}$. In order to always move towards $\mathbf{F}_{p,goal}$, we choose the final plastic deformation to be the point on the segment from $\mathbf{F}_p$ to $\mathbf{F}_{p,trial}$ which is closest to $\mathbf{F}_{p,goal}$. This computation is actually performed on the logarithms of each $\mathbf{F}_p$ after removing the world rotation. Since the mapping from rest to goal state will rarely preserve volume locally, volume preservation should not be used during the elastic projection step.

Allowing some flexibility in the plastic flow allows the deformation to pick up additional fine detail not present in the goal state. For example, Figure 9 shows a plastic sphere pulled through interlocking gears. The sphere is controlled towards the flattened disk shape. In particular, the goal state does not include the teeth marks present in the final state of the sphere. A more obvious example of control is shown in Figure 9. Both of these examples used the simple yield criterion $\| \log \mathbf{F}_e \| \leq \gamma$.

8 Thin Shells and Cloth

The diagonalized framework is readily extended to handle the in-plane behavior of triangles for modeling thin shells and cloth (see Figure 6). Here, $\mathbf{F}$ is a 3×2 matrix decomposed as $\mathbf{F} = \mathbf{U}\hat{\mathbf{F}}\mathbf{V}^T$ where $\mathbf{U}$ is a 3×2 matrix with orthonormal columns, $\hat{\mathbf{F}}$ is a 2×2 diagonal matrix, and $\mathbf{V}$ is a 2×2 rotation matrix. Everything else follows in a straightforward manner.

Inversion does not occur for freely moving thin shells and cloth, since an "inverted" triangle is indistinguishable from a triangle that has been rotated 180° out of plane. However, when triangles degenerate to lines or points special care *is* needed. Moreover, when a shell approximates a two dimensional surface such as during surface mesh generation (see e.g. [7]), "inversion" can occur. That is, a triangle can be tested for inversion by considering the sign of the dot product between its face normal and a known approximation to the surface normal at the center of the triangle. If this sign is negative, the triangle can

be considered inverted, and the signs of the entries of $\hat{\mathbf{F}}$ can be corrected as before. Thus, the triangle acts to uninvert by flipping the direction of its face normal.

For bending forces, we use the formulation of [35], which is similar to that of [36]. The bending model is independent of the in-plane model, and in-plane plasticity is analogous to the three dimensional case. To allow plastic bending, we apply the plastic flow algorithm to the rest angles between each pair of adjacent triangles. An example of a shell simulation showing both in-plane and bending plasticity is shown in Figure 10.

9 Time-Stepping and Collision Handling

We use the Newmark time-stepping scheme of [35] with explicit integration for the elastic forces and implicit integration for the damping forces. Treating only the damping forces implicitly removes the strict quadratic time step restriction required by fully explicit schemes without introducing the extra artificial damping characteristic of fully implicit schemes. As most damping models are linear in the velocities with a positive definite, symmetric Jacobian, the implicit integration can be implemented using a fast conjugate gradient solver.

We modify [35]'s scheme slightly to improve the handling of rigid body collisions. Specifically, we use the velocity from the last implicit update as input to the rigid body collision algorithm, and use constraints in the velocity update to prevent motion in the direction normal to the rigid body for points experiencing a collision. The resulting algorithm to move from step n to $n+1$ is as follows:

- $\tilde{v}^{n+1/2} = v^n + \frac{\Delta t}{2} a(t^n, x^n, \tilde{v}^{n+1/2})$
- $\tilde{x}^{n+1} = x^n + \Delta t \tilde{v}^{n+1/2}$
- Process rigid body collisions using $\tilde{x}^{n+1}$ and v^n, producing final positions x^{n+1} and modified velocities $\tilde{v}^n$.
- $v^{n+1} = \tilde{v}^n + \Delta t \left(a(t^n, x^n, \tilde{v}^n) + a(t^{n+1}, x^{n+1}, v^{n+1}) \right) / 2$

Note that the last line is exactly the trapezoidal rule applied to the velocities. This algorithm supports a variable time step with second order accuracy and monotone behavior. Since the positions change only in lines 2 and 3 of the algorithm, we can compute $\mathbf{F}$ and its diagonalization only once per time step after step 3. Plastic flow is also computed at this time. With this optimization, the cost of diagonalization becomes *negligible* compared to the cost of the implicit velocity updates.

The rigid body collision processing is based on the algorithm of [35]. We represent rigid bodies as implicit surfaces, which simplifies collision detection. Each such node is projected to the surface of the object, and its normal velocity is set to that of the object if it is not already moving away from it. We incorporate friction by changing the relative tangential velocity $v_{T,rel}$ to

$$v_{T,rel}^{new} = \max\left(0, 1 - \mu\frac{\Delta v_N + \Delta x_N/\Delta t}{|v_{T,rel}|}\right) v_{T,rel}$$

where Δx_N and Δv_N are the changes in position and normal velocity from the projection step. The $\Delta x_N/\Delta t$ term ensures that the particle will experience the correct friction for the change in position imparted by the object. This term was not considered in [35].

Any node involved in a collision is flagged, and its normal velocity is held fixed during the final trapezoidal rule step. Enforcing normal velocities of colliding particles via constraints during the velocity update further increases the stability of the collision scheme, since it allows a nonlocal response to collision. This strategy is similar to that proposed in [38], who implemented rigid body collisions in their fully implicit scheme via constraints in the conjugate gradient solver.

Since projecting points to the surface of an object tends to crush elements, the ability to handle flat or inverted elements is essential to enable the use of reasonable time steps. Also, since the rigid body collision algorithm is applied to surface vertices only, not surface triangles, it is useful to apply the algorithm to the interior points as well as the surface points, to prevent small rigid bodies from slipping between surface points into the interior of the object. The importance of this increases for very soft objects, as very soft surface triangles can easily expand and pass around even moderately sized obstacles.

For self-collisions we extract the boundary surface and apply the cloth collision algorithm of [37]. This algorithm is applied "outside" of the time-stepping algorithm outlined above. Although a surface-only collision algorithm does not prevent the interior of the object from extending outside its boundary, our method has no difficulty with this inversion and only the surface is needed for rendering.

10 Examples

We used the algorithm of [7] to generate the tetrahedral meshes used in this paper. Even without preconditioning in the CG solver, computation times were generally under 20 minutes per frame for the largest meshes. Of course, coarser

meshes can be simulated in just a few minutes a frame. For example, the torus simulation in Figure 2 ran at around 10 to 20 seconds per frame with the 115K element mesh, and .5 to 1 second per frame with an 11K element mesh. All the simulations involved large numbers of inverted elements: a typical frame from the Buddha simulation in Figure 7 had about 29K inverted tetrahedrons out of a total of 357K tetrahedrons, or about 8% of the mesh.

The Buddha with cape example in Figure 6 was simulated in two layers, with one-way coupling from the Buddha to the cloth using the collision processing algorithm from section 9. We used the exact triangulated surface geometry of the Buddha in order for the cloth to resolve the many features of the Buddha mesh. To evaluate the Buddha as an implicit surface at a cloth vertex $\mathbf{v}$, we find the closest point $\mathbf{p}$ to $\mathbf{v}$ on the Buddha surface (which may lie on a vertex, edge, or face) and define the "normal" at $\mathbf{v}$ to be in the direction from $\mathbf{v}$ to $\mathbf{p}$ or $\mathbf{p}$ to $\mathbf{v}$, whichever points outwards.

The hexahedral meshes shown in Figures 3 and 8 were cut out of regular cubic grids in the obvious way. For self-collision handling and rendering, we divide each boundary quadrilateral into two triangles. This adds no new degrees of freedom, and does not effect the hexahedral force computation.

11 Comparison of Tetrahedral and Hexahedral Elements

Given the ability to robustly handle inverted elements, it becomes possible to use all vertex degrees of freedom in order to represent deformation without danger of breaking the simulation. Therefore, we can compare the efficiency of tetrahedral and hexahedral elements simply by comparing the effort required to evaluate the force on each vertex. This can be further broken down into the number of quadrature points per vertex and the cost of each quadrature point. A standard cubic grid of hexahedrons has approximately one element per vertex, and 8 quadrature points per element, or about 8 quadrature points per vertex. Dividing each hexahedron into 5 tetrahedra with a Freudenthal cut produces 5 quadrature points per vertex. The regular BCC tetrahedral lattice used in [7] has 4 tetrahedra for each face in a cubic grid and doubles the number of vertices by adding the center of each cube, resulting in 6 quadrature points per vertex. In both of these examples, the number of quadrature points per vertex is larger for hexahedral meshes than for tetrahedral meshes. Since the work required per quadrature point is also larger for hexahedra due to the use of 8×3 matrices, tetrahedral elements are significantly faster per vertex than hexahedral elements.

12 Discussion

We stress here that element inversion does not imply that mass or even volume is necessarily lost. In fact, there are precedents for our approach in the finite element literature. When considering incompressible or nearly incompressible materials, linear tetrahedral elements suffer from severe volumetric and strain locking precluding their use. The problem occurs because tetrahedra have limited flexibility under the constraint that their volume has to be preserved (or almost preserved), especially when they are connected into a mesh of elements that all have this same volume preservation constraint. [50] partially alleviate this problem by allowing material to flow between elements so that the overall volume can be preserved without preserving the volume of each individual element. Another very interesting approach is the F-bar approach of [51] (see also [52]). The F-bar approach defines a modified deformation gradient $\bar{\mathbf{F}}$, which replaces the volume change in each element with the average volume change over a patch of several elements. This is similar in spirit to composite element approaches, see e.g. [53,54]. The F-bar approach allows individual elements to change volume as much as they want as long as the volume of the patch is preserved, significantly alleviating difficulties with locking.

Element inversion can be viewed in the context of incompressibility and locking. Restricting individual linear tetrahedra to a non-inverted state is too restrictive for some deformations, and it would be better to consider a patch of elements as in the F-bar technique. That is, one could imagine a method that allows an individual tetrahedron to invert as long as a larger patch of tetrahedra that contain the inverted element does not invert. Then mass (or volume) is not lost, but just transferred to other tetrahedra in the patch.

13 Conclusions

We have presented a new method for modifying an elastic constitutive model to behave robustly for inverted elements, which works by carefully diagonalizing the deformation mapping prior to computing forces. Examples were presented to demonstrate that this algorithm works well for volumetric and thin shell simulations involving degeneracy, complex geometries, anisotropic constitutive models, plasticity with and without control, ductile fracture, and coupling between different types of deformable objects. Although we originally presented this method for tetrahedra (only) in [11], here we generalized its application to quadrature points in arbitrary elements presenting specific details and simulations for the case of hexahedra.

14 Acknowledgement

Research supported in part by an ONR YIP award and a PECASE award (ONR N00014-01-1-0620), a Packard Foundation Fellowship, a Sloan Research Fellowship, ONR N00014-03-1-0071, ARO DAAD19-03-1-0331, NSF IIS-0326388, NSF ACI-0323866, NSF ITR-0205671 and NIH U54-GM072970. In addition, G. I. and J. T. were supported in part by NSF Graduate Research Fellowships.

We would also like to thank Mike Houston for providing computing resources used for rendering.

References

[1] C. Hirt, A. Amsden, J. Cook, An arbitrary Lagrangian-Eulerian computing method for all flow speeds, J. Comput. Phys. 135 (1974) 227–253.

[2] G. Camacho, M. Ortiz, Adaptive Lagrangian modelling of ballistic penetration of metallic targets, Comput. Meth. in Appl. Mech. and Eng. 142 (1997) 269–301.

[3] H. Espinosa, P. Zavattieri, G. Emore, Adaptive FEM computation of geometric and material nonlinearities with application to brittle failure, Mech. Materials 29 (1998) 275–305.

[4] G. Bessette, E. Becker, L. Taylor, D. Littlefield, Modeling of impact problems using an h-adaptive, explicit lagrangian finite element method in three dimensions, Comput. Meth. in Appl. Mech. and Eng. 192 (2003) 1649–1679.

[5] P. Vachal, R. Garimella, M. Shashkov, Untangling of 2D meshes in ALE simulation, J. Comput. Phys. 196 (2004) 627–644.

[6] J. Escobar, E. Rodríguez, R. Montenegro, G. Montero, J. González-Yuste, Simultanous untangling and smoothing of tetrahedral meshes, Comput. Meth. in Appl. Mech. and Eng. 192 (2003) 2775–2787.

[7] N. Molino, R. Bridson, J. Teran, R. Fedkiw, A crystalline, red green strategy for meshing highly deformable objects with tetrahedra, in: 12th Int. Meshing Roundtable, 2003, pp. 103–114.

[8] R. Kautzman, A. Maiolo, D. Griffin, A. Bueker, Jiggly bits and motion retargetting: Bringing the motion of hyde to life in van helsing with dynamics, in: SIGGRAPH 2004 Sketches & Applications, ACM Press, 2004.

[9] M. Teschner, B. Heidelberger, M. Muller, M. Gross, A versatile and robust model for geometrically complex deformable solids, in: Proc. Computer Graphics International, 2004.

[10] M. Muller, M. Gross, Interactive virtual materials, in: Graph. Interface, 2004.

[11] G. Irving, J. Teran, R. Fedkiw, Invertible finite elements for robust simulation of large deformation, in: Proc. of the ACM SIGGRAPH/Eurographics Symp. on Comput. Anim., 2004, pp. 131–140.

[12] D. Terzopoulos, J. Platt, A. Barr, K. Fleischer, Elastically deformable models, Comput. Graph. (Proc. SIGGRAPH 87) 21 (4) (1987) 205–214.

[13] D. Terzopoulos, K. Fleischer, Modeling inelastic deformation: viscoelasticity, plasticity, fracture, Comput. Graph. (SIGGRAPH Proc.) (1988) 269–278.

[14] D. Terzopoulos, K. Fleischer, Deformable models, The Visual Computer (4) (1988) 306–331.

[15] J.-P. Gourret, N. Magnenat-Thalmann, D. Thalmann, Simulation of object and human skin deformations in a grasping task, Comput. Graph. (SIGGRAPH Proc.) (1989) 21–30.

[16] D. Chen, D. Zeltzer, Pump it up: Computer animation of a biomechanically based model of muscle using the finite element method, Comput. Graph. (SIGGRAPH Proc.) (1992) 89–98.

[17] G. Picinbono, H. Delingette, N. Ayache, Non-linear and anisotropic elastic soft tissue models for medical simulation, in: IEEE Int. Conf. Robot. and Automation, 2001.

[18] Q. Zhu, Y. Chen, A. Kaufman, Real-time biomechanically-based muscle volume deformation using FEM, Comput. Graph. Forum 190 (3) (1998) 275–284.

[19] G. Hirota, S. Fisher, A. State, C. Lee, H. Fuchs, An implicit finite element method for elastic solids in contact, in: Proc. of Computer Animation, 2001, pp. 136–146.

[20] J. O'Brien, J. Hodgins, Graphical modeling and animation of brittle fracture, in: Proc. SIGGRAPH 99, Vol. 18, 1999, pp. 137–146.

[21] J. O'Brien, A. Bargteil, J. Hodgins, Graphical modeling of ductile fracture, ACM Trans. Graph. (SIGGRAPH Proc.) 21 (2002) 291–294.

[22] G. Yngve, J. O'Brien, J. Hodgins, Animating explosions, in: Proc. SIGGRAPH 2000, Vol. 19, 2000, pp. 29–36.

[23] M. Muller, L. McMillan, J. Dorsey, R. Jagnow, Real-time simulation of deformation and fracture of stiff materials, in: Comput. Anim. and Sim. '01, Proc. Eurographics Workshop, Eurographics Assoc., 2001, pp. 99–111.

[24] G. Debunne, M. Desbrun, M. Cani, A. Barr, Dynamic real-time deformations using space & time adaptive sampling, in: Proc. SIGGRAPH 2001, Vol. 20, 2001, pp. 31–36.

[25] M. Muller, J. Dorsey, L. McMillan, R. Jagnow, B. Cutler, Stable real-time deformations, in: ACM SIGGRAPH Symp. on Comput. Anim., 2002, pp. 49–54.

[26] S. Capell, S. Green, B. Curless, T. Duchamp, Z. Popović, Interactive skeleton-driven dynamic deformations, ACM Trans. Graph. (SIGGRAPH Proc.) 21 (2002) 586–593.

[27] S. Capell, S. Green, B. Curless, T. Duchamp, Z. Popović, A multiresolution framework for dynamic deformations, in: ACM SIGGRAPH Symp. on Comput. Anim., ACM Press, 2002, pp. 41–48.

[28] J. Teran, S. Blemker, V. Ng, R. Fedkiw, Finite volume methods for the simulation of skeletal muscle, in: Proc. of the 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim., 2003, pp. 68–74.

[29] D. James, D. Pai, DyRT: Dynamic response textures for real time deformation simulation with graphics hardware, ACM Trans. Graph. (SIGGRAPH Proc.) 21 (2002) 582–585.

[30] D. James, K. Fatahalian, Precomputing interactive dynamic deformable scenes, ACM Trans. Graph. (SIGGRAPH Proc.) 22 (2003) 879–887.

[31] B. Palmerio, An attraction-repulsion mesh adaption model for flow solution on unstructured grids, Comput. and Fluids 23 (3) (1994) 487–506.

[32] D. Bourguignon, M. P. Cani, Controlling anisotropy in mass-spring systems, in: Eurographics, Eurographics Assoc., 2000, pp. 113–123.

[33] L. Cooper, S. Maddock, Preventing collapse within mass-spring-damper models of deformable objects, in: The 5th Int. Conf. in Central Europe on Comput. Graphics and Vis., 1997.

[34] O. Etzmuss, M. Keckeisen, W. Strasser, A fast finite element solution for cloth modelling, in: Pacific Graph., 2003, pp. 244–251.

[35] R. Bridson, S. Marino, R. Fedkiw, Simulation of clothing with folds and wrinkles, in: Proc. of the 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim., 2003, pp. 28–36.

[36] E. Grinspun, A. Hirani, M. Desbrun, P. Schroder, Discrete shells, in: Proc. of the 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim., 2003, pp. 62–67.

[37] R. Bridson, R. Fedkiw, J. Anderson, Robust treatment of collisions, contact and friction for cloth animation, ACM Trans. Graph. (SIGGRAPH Proc.) 21 (2002) 594–603.

[38] D. Baraff, A. Witkin, Large steps in cloth simulation, in: Proc. SIGGRAPH 98, 1998, pp. 1–12.

[39] K.-J. Choi, H.-S. Ko, Stable but responsive cloth, ACM Trans. Graph. (SIGGRAPH Proc.) 21 (2002) 604–611.

[40] E. Grinspun, P. Krysl, P. Schroder, CHARMS: A simple framework for adaptive simulation, ACM Trans. Graph. (SIGGRAPH Proc.) 21 (2002) 281–290.

[41] D. Baraff, A. Witkin, M. Kass, Untangling cloth, ACM Trans. Graph. (SIGGRAPH Proc.) 22 (2003) 862–870.

[42] T. Hughes, The Finite Element Method: Linear Static and Dynamic Finite Element Analysis, Prentice Hall, 1987.

[43] J. Bonet, R. Wood, Nonlinear continuum mechanics for finite element analysis, Cambridge University Press, Cambridge, 1997.

[44] J. Teran, E. Sifakis, S. Salinas-Blemker, V. Ng-Thow-Hing, C. Lau, R. Fedkiw, Creating and simulating skeletal muscle from the visible human data set, IEEE Trans. on Vis. and Comput. Graph. 11 (3) (2005) 317–328.

[45] F. Armero, E. Love, An arbitrary lagrangian-eulerian finite element method for finite strain plasticity, Int. J. Num. Meth. Eng. 57 (2003) 471–508.

[46] N. Molino, J. Bao, R. Fedkiw, A virtual node algorithm for changing mesh topology during simulation, ACM Trans. Graph. (SIGGRAPH Proc.) 23 (2004) 385–392.

[47] N. Foster, D. Metaxas, Controlling fluid animation, in: Computer Graphics International 1997, 1997, pp. 178–188.

[48] J. Popović, S. Seitz, M. Erdmann, Z. Popović, A. Witkin, Interactive manipulation of rigid body simulations, ACM Trans. Graph. (SIGGRAPH Proc.) 19 (2000) 209–217.

[49] A. Treuille, A. McNamara, Z. Popovic, J. Stam, Keyframe control of smoke simulations, ACM Trans. Graph. (SIGGRAPH Proc.) 22 (2003) 716–723.

[50] B. Boroomand, B. Khalilian, On using linear elements in incompressible plane strain problems: a simple edge based approach for triangles, Int. J. Num. Meth. Eng. 61 (2004) 1710–1740.

[51] E. de Souza Neto, F. A. Pires, D. Owen, F-bar-based linear triangles and tetraehdra for finite strain analysis of nearly incompressible solids. part i: formulation and benchmarking, Int. J. Num. Meth. Eng. 62 (2005) 353–383.

[52] F. A. Pires, E. de Souza Neto, D. Owen, On the finite element prediction of damage growth and fracture initiation in finitely deforming ductile materials, Comput. Meth. in Appl. Mech. and Eng. 193 (2004) 5223–5256.

[53] Y. Guo, M. Ortiz, T. Belytschko, E. Repetto, Triangular composite finite elements, Int. J. Num. Meth. Eng. 47 (2000) 287–316.

[54] P. Thoutireddy, J. Molinari, E. Repetto, M. Ortiz, Tetrahedral composite finite elements, Int. J. Num. Meth. Eng. 53 (2002) 1337–1351.

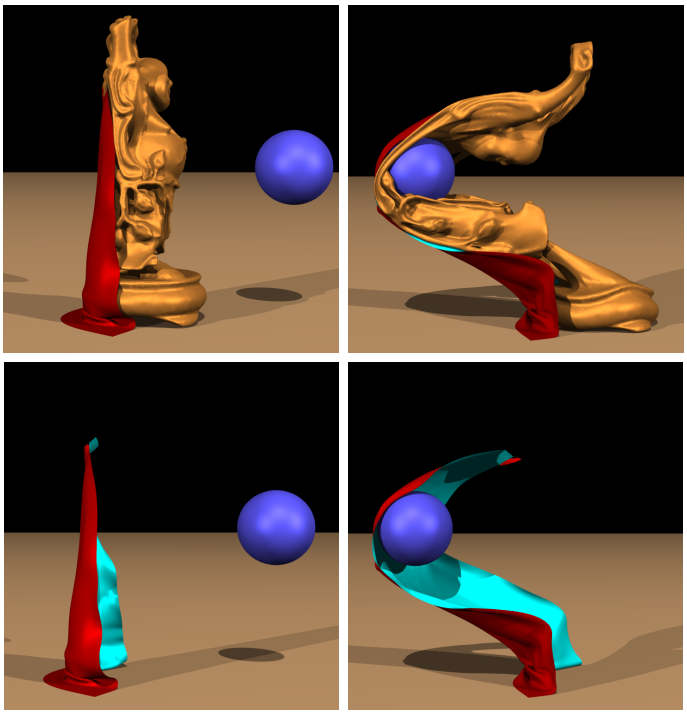

Fig. 6. A deformable Buddha with a cape undergoing large deformation when hit by a ball (top). The same with the Buddha removed to illustrate the deformation (bottom). (Cape - 84K triangles, Buddha - 357K tetrahedrons)

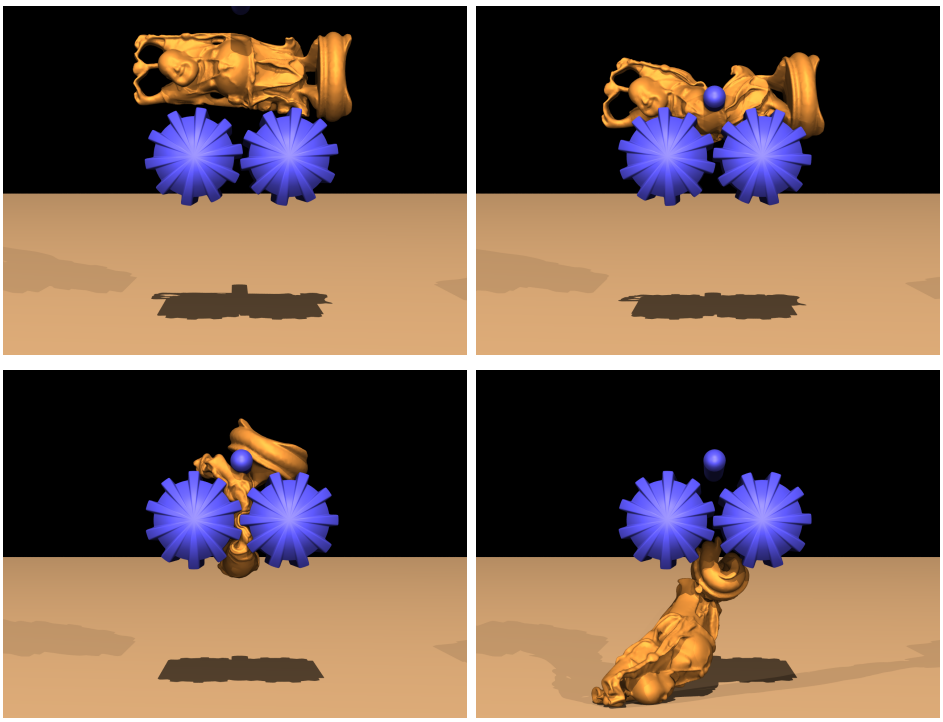

Fig. 7. A volumetric Buddha model is pushed down with a cylinder and pulled between rigid interlocking gears, then recovers its shape elastically. (300K tetrahedrons)

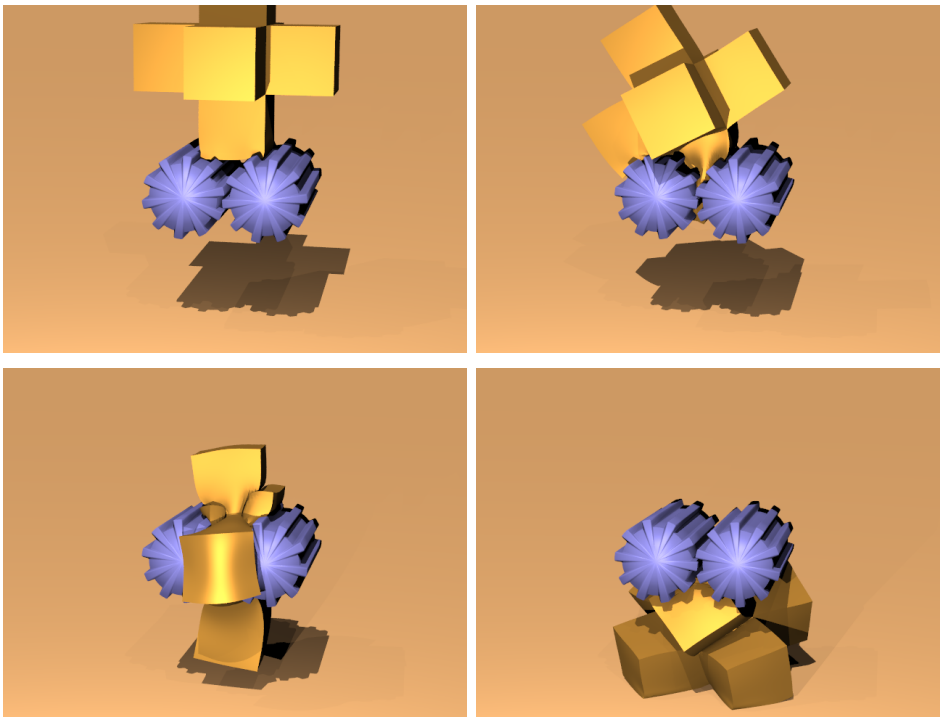

Fig. 8. A hexahedralized volume pulled between rigid interlocking gears. (56K hexahedrons)

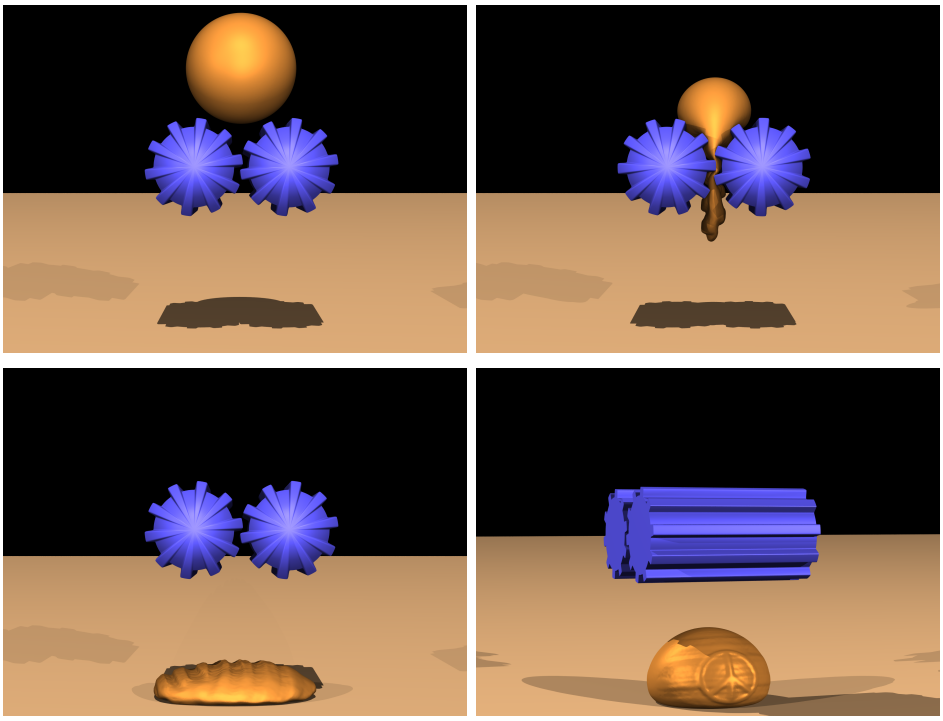

Fig. 9. A plastic sphere controlled towards a flattened disk shape is pulled through rigid interlocking gears (upper left, upper right, lower left). A more obvious example of plasticity control (lower right).

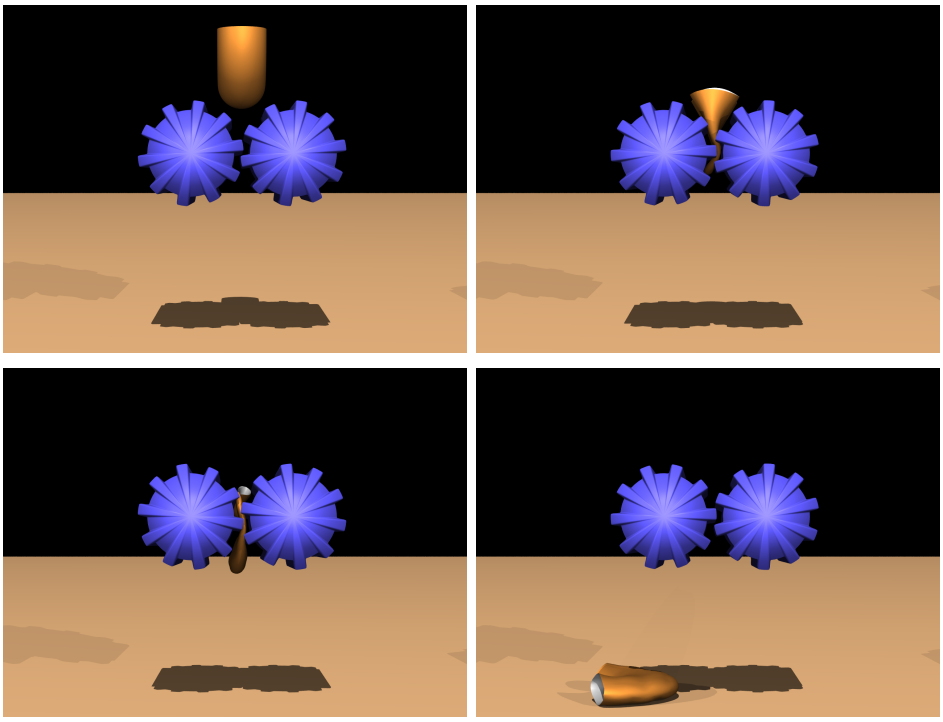

Fig. 10. Half of a torus (shell) simulated with in-plane and bending plasticity. (3.5K triangles).

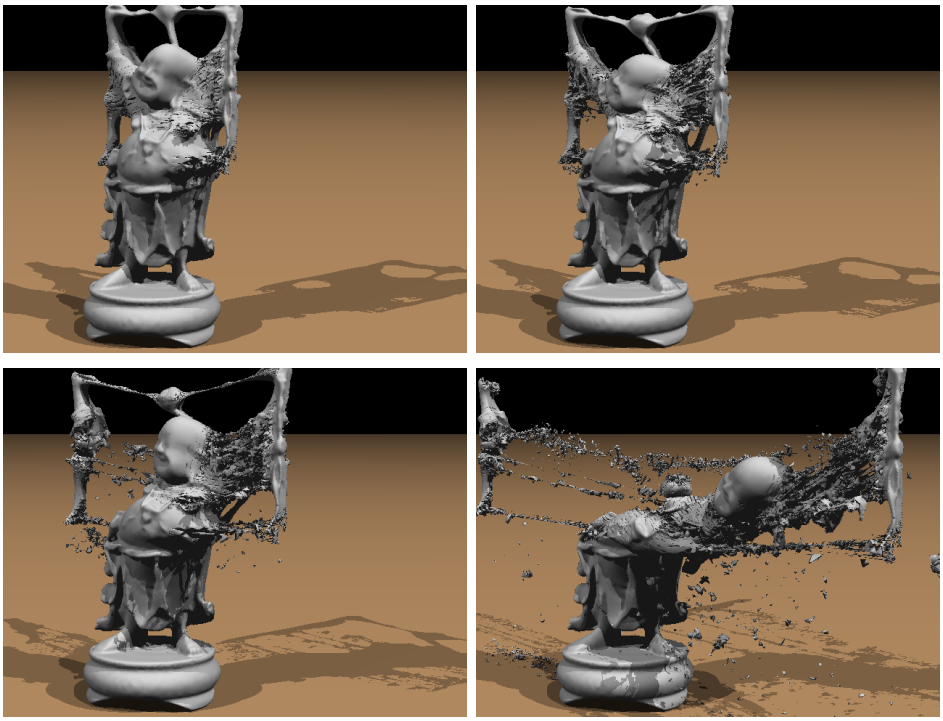

Fig. 11. A Buddha undergoing ductile fracture. (300K tetrahedrons)