Fracturing Rigid Materials

Zhaosheng Bao, Jeong-Mo Hong, Joseph Teran, and Ronald Fedkiw, *Member, IEEE*

***Abstract*— We propose a novel approach to fracturing (and denting) brittle materials. To avoid the computational burden imposed by the stringent time step restrictions of explicit methods or with solving nonlinear systems of equations for implicit methods, we treat the material as a fully rigid body in the limit of infinite stiffness. In addition to a triangulated surface mesh and level set volume for collisions, each rigid body is outfitted with a tetrahedral mesh upon which finite element analysis can be carried out to provide a stress map for fracture criteria. We demonstrate that the commonly used stress criteria can lead to arbitrary fracture (especially for stiff materials) and instead propose the notion of a time averaged stress directly into the FEM analysis. When objects fracture, the virtual node algorithm provides new triangle and tetrahedral meshes in a straightforward and robust fashion. Although each new rigid body can be rasterized to obtain a new level set, small shards can be difficult to accurately resolve. Thus we propose a novel collision handling technique for treating both rigid bodies and rigid body thin shells represented by only a triangle mesh.**

***Index Terms*— fracture, rigid bodies, finite element analysis.**

I. Introduction

WETHER blowing up buildings, shattering glass or destroying spaceships, fracture is ubiquitous in the movie industry. Of course, numerical simulations are typically a safer alternative for creating these effects. The least user intensive and most realistic algorithms use modern continuum mechanics to automatically determine fracture patterns via externally induced stress fields in response to interactions with explosions, projectiles, etc.

Many of these effects require the fracturing of nearly rigid materials, which can be difficult to treat with physically based approaches. Explicit time integration schemes suffer from stringent time step restrictions, especially if fracture and subsequent remeshing generate poorly conditioned or even sliver elements [1]. Quasistatic and implicit time integration schemes (see e.g. [2]–[4]) alleviate these restrictions, but the associated nonlinear systems have a computational burden of their own. Moreover, sliver elements can lead to poor conditioning of the linearized subproblems. The difficulties are exacerbated, since costly remeshing algorithms are typically the only way to combat poorly conditioned elements [1]. Rigid body models naturally address these difficulties, but fail to provide the deformation and internal stress states required for realistic fracture calculations. In this paper, we propose a hybrid approach using rigid body simulation for efficient dynamic evolution and fully linearized, small displacement finite elements to determine rich and realistic fracture patterns. A similar approach was taken in [5], although they used nonlinear FEM requiring Newton-Raphson iteration, as well as Gauss-Seidel smoothing. This general approach of mixing rigid body simulation with finite element analysis was also stressed in [6] (see also [7]) where a deformable model was used for rigid body contact handling.

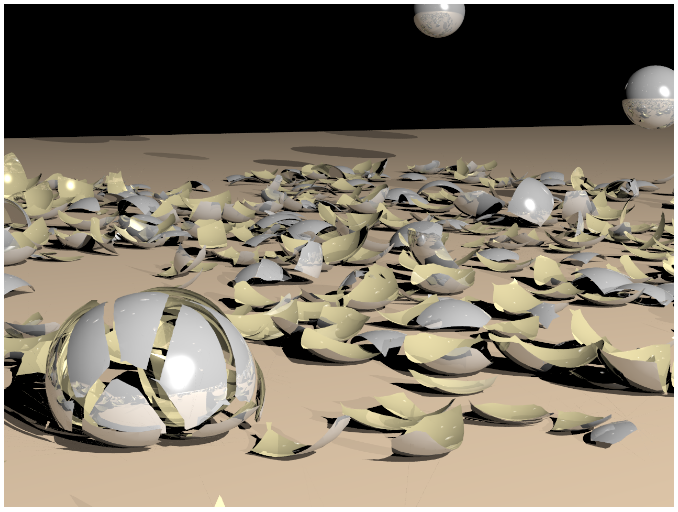


Fig. 1. Shattering ornaments are modeled as rigid thin shells. Fracture patterns result from instantaneous quasistatic FEM analysis.

We simulate all objects as rigid bodies endowed with a triangulated surface mesh, a level set volume for inside/outside information, and a tetrahedron mesh for finite element calculations. We use the impulse based rigid body simulation method outlined in [8], although any other robust rigid body solver could be used (e.g. [9]) with appropriate modifications. After fracture lines

Zhaosheng Bao, Jeong-Mo Hong, and Ron Fedkiw are with the Department of Computer Science, Stanford University. Ron Fedkiw is also with Industrial Light and Magic. Joseph Teran is with the Courant Institute, New York University. Email: {jbao,jeongmo,fedkiw}@cs.stanford.edu. teran@cims.nyu.edu

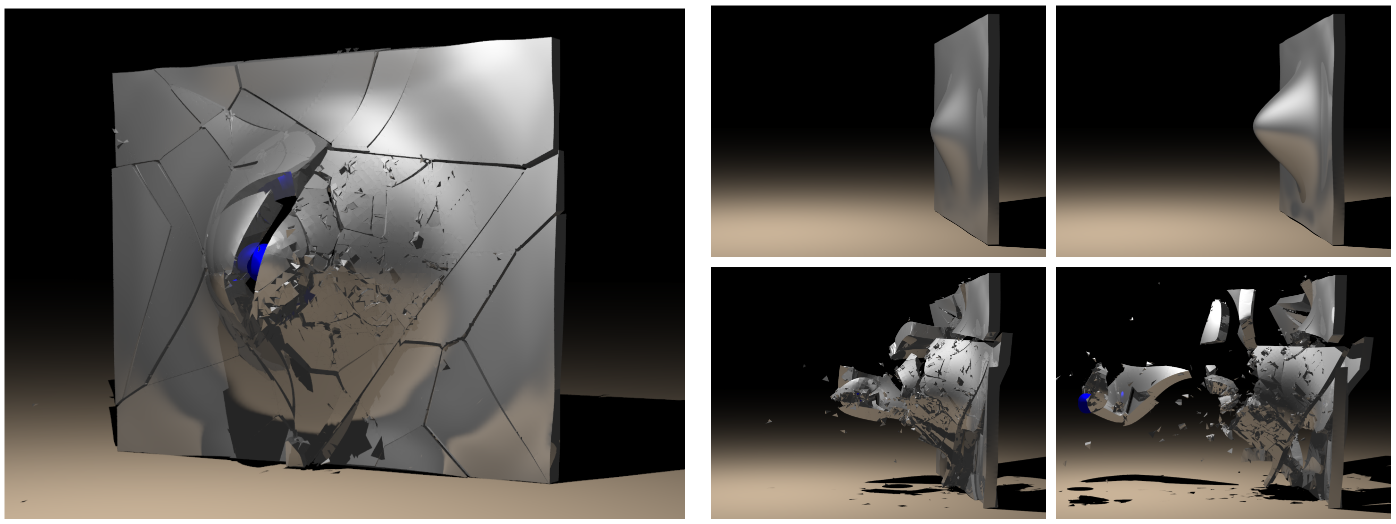


Fig. 2. A ballistic object impacts a metallic slab causing intricate ductile fracture patterns.

are determined, the virtual node algorithm [10] is used to automatically cut the tetrahedral mesh and form a new embedded triangulated surface. Then a flood fill algorithm is used to determine when embedded cracks cause a new rigid body to form. While the virtual node algorithm automatically gives us a new triangulated surface and tetrahedralized volume for each new rigid body, dynamic properties (such as mass, inertia, velocity, etc.) and new level set volumes are calculated on-the-fly.

Accurate finite element analysis of our rigid bodies is of the utmost importance, since it is subsequently used to generate interesting fracture patterns. Under the small deformations typical of stiff material, quasi-static finite element models are appropriate but suffer from the existence of a null space. This issue was avoided in [5] by anchoring (freezing) tetrahedra not local to the collision event. Unfortunately, this adversely effects the plausibility of the computed stress and resulting fracture patterns. In the quest for better fracture patterns, we instead fix the null space issue directly by identifying and eliminating it during our conjugate gradient solution procedure. Moreover, we point out (below) that stress can vary arbitrarily during collisions, especially for stiff materials (such as rigid bodies), and instead propose a method based on impulses leading to a more appropriate time averaged stress.

Our simulations generate new geometry at a range of scales, and it quickly becomes evident that it is not efficient to generate a volumetric level set representation for each small shard. Thus, we devised a new approach to rigid body simulation that allows for arbitrarily thin rigid body shells composed only of triangles. This is more efficient than simulating stiff shells with implicit time stepping as in [3]. In particular, we devised a new robust collision handling technique (based in part on [11]) for treating collisions between these thin shell rigid bodies. Other work on thin shell fracture includes [12] which uses vertex budging to allow fracture lines to pass more smoothly through a triangle mesh. Thin shell fracture has also recently been addressed for point based methods [13] (see [14] for volumetric examples), but these methods currently lack robust self-collision handling especially in the context of fracture and small shards.

II. Related Work

[15] pioneered deformable models in graphics including early work on fracture [16], [17] where cloth was torn. Finite element methods have remained popular, and there are many examples of their use: a hand grasping a ball [18], virtual surgery [19], muscles [20], [21], etc. Other interesting works include adaptive frameworks [22]–[24], rotation based approaches [25]–[28], and precomputed data driven models [29]–[31]. Some of the most recent work on deformable models includes the meshless approach of [32]. Also quite interesting, and relevant to our approach is the particle based approach to granular materials and rigid bodies [33].

The simplest fracture models are built by simply breaking the connection between elements when the forces are high as in [34]–[36]. For example, [37] replaced stiff springs with distance preserving linear constraints while using associated Lagrange multipliers to compute forces and determine fracture patterns. Although it can be difficult to subdivide elements, many strategies have been proposed (especially for tetrahedra, see e.g. [38]). Unfortunately, the resulting mesh can be ill-conditioned and difficult to simulate, see e.g. [39]. A

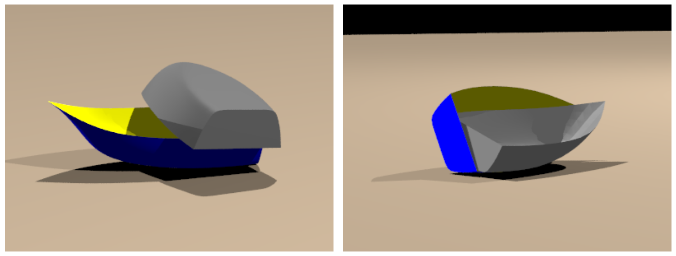


Fig. 3. Our thin shell rigid body algorithm exhibits robust edge/edge collision of two infinitely thin rigid boats made entirely of triangles.

seminal computer graphics paper for three-dimensional fracture is [1], where the authors proposed strategies for splitting and simulating tetrahedral elements with explicit time stepping. Although faced with lengthy simulation times, their results were impressive. This work was later extended to ductile fracture [40] and explosions [41] (also note the related blast wave fracture work of [42]).

In a vein similar to our own, [5] treated objects as rigid bodies *between* collisions, and used static finite element analysis techniques *during* collisions. They used the principal stress components to separate tetrahedra, occasionally refining large tetrahedra before splitting along element boundaries. Although they claim that static analysis can only be applied for supported objects (and thus they anchor objects to the ground), we show that this is not true if the null space is addressed. [43] extended this work adding the ability to simulate and fracture more complex geometry embedded in a uniform mesh of cubes. [27] similarly embeds a higher resolution surface mesh into a lower resolution simulation mesh, proposing methods for hole filling during fracture. [10] removed the difficulties associated with continuous remeshing, instead proposing a virtual node algorithm that automatically separates the mesh preserving large tetrahedral elements while avoiding ill-conditioned slivers. Recent point based approaches have also been extended to handle fracture [13], [14].

III. Thin Shell Rigid Body Model

We simulate each object as a rigid body using the impulse based framework of [8], including both an explicit triangulated surface and an implicit signed distance function for contact and collision. When considering very thin rigid bodies, especially those that can be generated during fracture (such as shards), grid resolution is critical for preventing thin bodies from passing completely through each other undetected unless an overly restrictive time step is taken. Thus, we propose a new algorithm for treating contact and collision, especially suited for thin

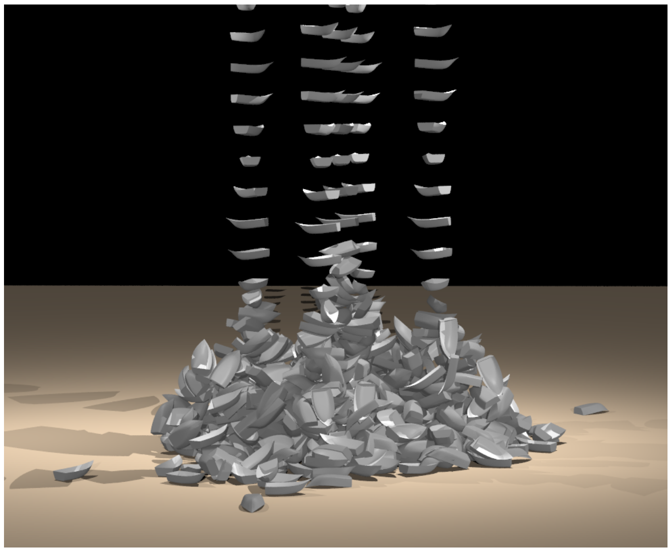


Fig. 4. The thin shell collision algorithm scales to large numbers of bodies as demonstrated by this pile of 1000 boats.

shell rigid bodies. Our new method does not require an implicit signed distance function, but works solely with a triangulated surface allowing us to handle thin shells with no interior or exterior region. Moreover, it is based on the cloth collision framework of [11] allowing us to take large time steps without missing any collisions between the triangles that represent the surface of our object.

In the first step of our thin shell approach, we allow the object to deform slightly in response to collisions. For example, suppose we start with a collision free triangulated surface at time n, and want to advance to time $n+1$. We first evolve the rigid bodies to their new positions, and then create a velocity for each particle on the triangulated surface that takes it from its time n position to its time $n+1$ position. Then this triangle soup of particle positions and velocities can be processed with the cloth collision scheme of [11] in order to find a collision free state at time $n+1$. Since the cloth collision algorithm is based on deformable bodies, the collision free state will generally be a deformed version of the rigid body. See Figure (5), top. However, at each subsequent time step, we target the undeformed rigid body state version of the triangulated surface when generating particle velocities for the cloth collision algorithm. See Figure (5), bottom. Thus, the deformed triangulated surface will again conform to the rigid body shape whenever collisions do not prevent it.

The cloth collision algorithm applies impulses between point/face and edge/edge pairs that collide [11]. We accumulate all of these impulses and use them to apply net translation and rotational collision impulses to

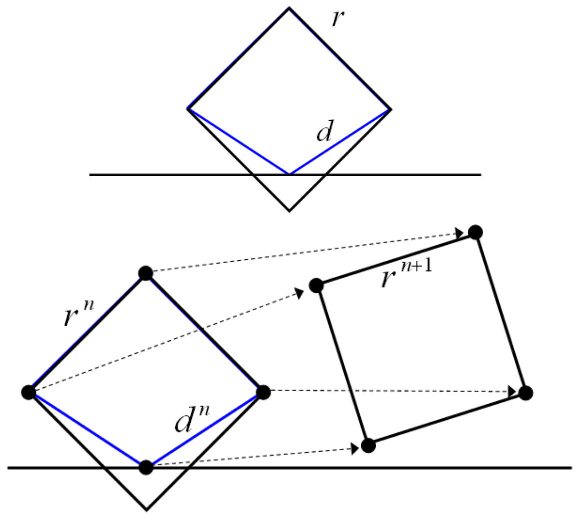


Fig. 5. First, collisions are resolved by deforming the bodies if necessary (top). Then in the next time step, a deformed state is retargeted towards an undeformed rigid body state (bottom).

the corresponding rigid bodies in their time n configuration. The rigid bodies are then integrated to the time $n+1$ configuration.

Collisions with other types of objects, such as those represented by level sets, can be incorporated by first applying these impulses to rigid bodies before executing the cloth collision handling step. Note that the notion of letting the triangulated surface drift off its rest state as a rigid body is similar to the collision handling treatment proposed in [10] for a triangulated surface that was embedded in a tetrahedral mesh. They too let nodes drift and subsequently retarget their desired positions every time step. Examples of our new method in action are shown in figures 3 and 4.

IV. Finite Element Analysis

Two novel aspects of our approach are our use of time averaged stress when determining fracture patterns, and the manner in which we handle the null space of the linearly elastic stiffness matrix. The stiffness matrix has a null-space consisting of linearized rigid motions that can be projected away with boundary conditions. However, we show how to process it without boundary conditions (or ad hoc anchoring) in a purely rigid fashion in the context of our rigid fracture algorithm. The large time steps admitted by our rigid body evolution assumption can lead to inaccurate fracture behavior, and our time averaged stress approach leads to fracture phenomena more consistent as the size of the time step varies. This section develops the notation required to fully describe these two novel aspects of our algorithm.

For thin shell rigid bodies, we can compute the stress directly on the (undeformed) triangulated surface used for collisions, while for volumetric rigid bodies we first construct a three-dimensional tetrahedral mesh using the algorithm from [45]. For nearly rigid materials, large stresses occur from very small changes in positions making the system ill-conditioned. This is typically remedied by using displacement $\vec{u} = \vec{x} - \vec{X}$ (from the rest position $\vec{X}$) instead of position $\vec{x}$ as the independent variable. Moreover, we never actually move the mesh, since the movement is visually negligible anyway. Instead, we just store a displacement on each node for the purpose of calculating stress.

For constant strain elements, $\vec{x} = \mathbf{F}\vec{X} + \vec{b}$, or in terms of displacements $\vec{u} = (\mathbf{F} - \mathbf{I})\vec{X} + \vec{b}$. The Jacobian of the displacements, $\mathbf{F}_u = \mathbf{F} - \mathbf{I}$, can be computed via $\mathbf{F}_u = \mathbf{D}_u\mathbf{D}_m^{-1}$ where $\mathbf{D}_u = (\vec{u}_1 - \vec{u}_0, \vec{u}_2 - \vec{u}_0, \vec{u}_3 - \vec{u}_0)$ and $\mathbf{D}_m = (\vec{X}_1 - \vec{X}_0, \vec{X}_2 - \vec{X}_0, \vec{X}_3 - \vec{X}_0)$. Here, the numerical subscripts refer to the individual nodes of a tetrahedron. For triangles, $\mathbf{F}_u$ and $\mathbf{D}_u$ are 3x2 matrices, and $\mathbf{D}_m$ is a 2x2 matrix in the two-dimensional material space of the triangle.

For small displacements, we linearize the Green strain $\mathbf{G} = 1/2\left(\mathbf{F}^T\mathbf{F} - \mathbf{I}\right)$ about $\vec{X}$ noting that $\mathbf{G}|_{\vec{X}} = 0$. The details are $\mathbf{G}|_{\vec{x}} = \mathbf{G}|_{\vec{x}} - \mathbf{G}|_{\vec{X}} = \dot{\mathbf{G}}|_{\vec{x}=\vec{X}}(\vec{x} - \vec{X}) = 1/2(\mathbf{F}^T\dot{\mathbf{F}} + \dot{\mathbf{F}}^T\mathbf{F})|_{\vec{x}=\vec{X}}(\vec{x} - \vec{X}) = 1/2(\dot{\mathbf{F}}|_{\vec{x}=\vec{X}}(\vec{x} - \vec{X}) + \dot{\mathbf{F}}|^T_{\vec{x}=\vec{X}}(\vec{x} - \vec{X})) = 1/2((\mathbf{F} - \mathbf{I}) + (\mathbf{F}^T - \mathbf{I}))$. Hence, we denote the linearized green strain as $\mathbf{G}_L = 1/2\left(\mathbf{F}_u + \mathbf{F}_u^T\right)$.

The first and second Piola-Kirkoff stresses are related via $\mathbf{P} = \mathbf{FS}$, and $\dot{\mathbf{P}}|_{\vec{x}=\vec{X}} = (\dot{\mathbf{F}}\mathbf{S} + \mathbf{F}\dot{\mathbf{S}})|_{\vec{x}=\vec{X}} = \dot{\mathbf{S}}|_{\vec{x}=\vec{X}}$ shows that the linearized $\mathbf{P}$ and linearized $\mathbf{S}$ are equivalent. For linear elasticity, $\mathbf{S} = 2\mu\mathbf{G} + \lambda tr(\mathbf{G})$ where μ and λ are Lamé coefficients. Hence, $\mathbf{P}_L = \mathbf{S}_L = \dot{\mathbf{S}}|_{\vec{x}=\vec{X}}(\vec{x} - \vec{X}) = 2\mu\dot{\mathbf{G}}|_{\vec{x}=\vec{X}}(\vec{x} - \vec{X}) + \lambda tr(\dot{\mathbf{G}}|_{\vec{x}=\vec{X}}(\vec{x} - \vec{X})) = 2\mu\mathbf{G}_L + \lambda tr(\mathbf{G}_L) = \mu(\mathbf{F}_u + \mathbf{F}_u^T) + \lambda tr(\mathbf{F}_u + \mathbf{F}_u^T)/2 = \mu(\mathbf{F}_u + \mathbf{F}_u^T) + \lambda tr\mathbf{F}_u$, which is a homogeneous function of $\vec{u}$. The Cauchy stress, $\sigma = \mathbf{FSF}^T/\det(\mathbf{F})$, is commonly used as a fracture criteria, and we have $\dot{\sigma}|_{\vec{x}=\vec{X}} = (\mathbf{F}\dot{\mathbf{S}}\mathbf{F}^T)/\det(\mathbf{F})\Big|_{\vec{x}=\vec{X}} = \dot{\mathbf{S}}|_{\vec{x}=\vec{X}}$. Thus, $\sigma_L = \mathbf{S}_L = \mathbf{P}_L$ and all three stresses are equivalent when linearized.

The force on the three of the four nodes is $[\vec{f}_1|\vec{f}_2|\vec{f}_3] = \mathbf{PB}_m$ where $\mathbf{B}_m = -1/2Cofactor(\mathbf{D}_m)$ for triangles and $\mathbf{B}_m = -1/6Cofactor(\mathbf{D}_m)$ for tetrahedra. The force on the remaining node is computed from Newton's third law. Since $\mathbf{B}_m$ is a constant matrix, the forces are linear in $\mathbf{P}$ which is linear in $\vec{u}$.

Under the quasistatic assumption, we solve for the state $\vec{x}$ where the net force on the object is zero, or

$$\vec{f} = \vec{f}_{int}(\vec{x}) + \vec{f}_{ext} = 0$$

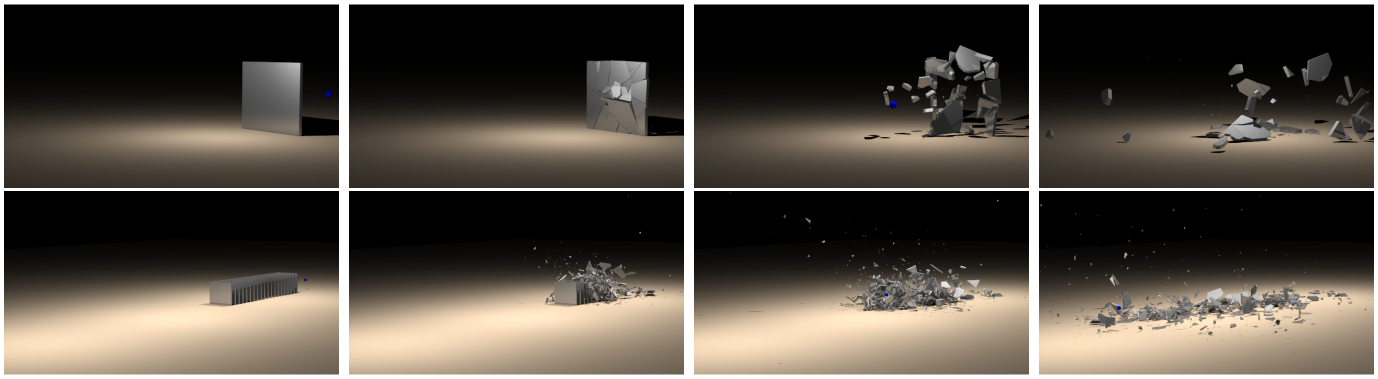


Fig. 6. The walls above demonstrate the realistic fracture of brittle materials that our method is capable of producing. The bottom sequence involves 20 walls and demonstrates the efficiency with which small shards are handled as the slabs are obliterated.

where the external forces come from collisions. Linearizing about $\vec{X}$, we have $\partial\vec{f}(\vec{x})/\partial\vec{x}\Big|_{\vec{x}=\vec{X}}(\vec{x}-\vec{X}) = \vec{f}(\vec{x}) - \vec{f}(\vec{X})$, and since $\vec{f}_{int}(\vec{X}) = 0$, $\vec{f}_{ext}$ is constant and $\vec{f}(\vec{x}) = 0$ by the quasistatic assumption, we have

$$\partial\vec{f}_{int}(\vec{x})/\partial\vec{x}\Big|_{\vec{x}=\vec{X}}(\vec{x}-\vec{X}) = -\vec{f}_{ext}.$$

Moreover, $\partial\vec{u}/\partial\vec{x} = I$ implies $(\partial\vec{f}_{int}(\vec{u})/\partial\vec{u})\vec{u} = -\vec{f}_{ext}$. $\vec{f}_{int}(\vec{u})$ is linear for linear elasticity and linear strain. So letting $\vec{f}_{int}(\vec{u}) = Mu$ implies that $(\partial\vec{f}_{int}(\vec{u})/\partial\vec{u})\vec{u} = Mu = \vec{f}_{int}(\vec{u})$ and we only need to solve the linear system $\vec{f}_{int}(\vec{u}) = -\vec{f}_{ext}$ with solution $\vec{u} = (-M)^{-1}\vec{f}_{ext}$.

V. Null Space Elimination

We use Conjugate Gradients to solve the linear system. However, M has a null space under net translation and linearized rotations. While $\mathbf{G}$ is invariant to rotation, i.e. identically zero when $\mathbf{F}$ is a rotation matrix, this is not true of $\mathbf{G}_L$. However, if $\mathbf{F}_u = \mathbf{R}_L(\theta) = \dot{\mathbf{R}}(0)\theta$ is a *linearized* rotation (see figure 7), differentiating $\mathbf{R}(\theta)\mathbf{R}^T(\theta) = \mathbf{I}$ gives $\dot{\mathbf{R}}(\theta)\mathbf{R}^T(\theta) + \mathbf{R}(\theta)\dot{\mathbf{R}}^T(\theta) = 0$ or $\dot{\mathbf{R}}(0) = -\dot{\mathbf{R}}^T(0)$ when $\theta = 0$. This means that $\mathbf{F}_u = -\mathbf{F}_u^T$ and $\mathbf{G}_L = 0$. The stretching and shearing artifacts characteristic of linearized Green strain under large deformation occur because the strain is unaware of them (i.e. unaware of linearized rotation). It is in the null space of M. We propose a procedure for removing this null space by projecting it out of the displacements before the line search performed in each iteration of conjugate gradient. We note that this is similar to how [2] performs the filtering operation before the line search.

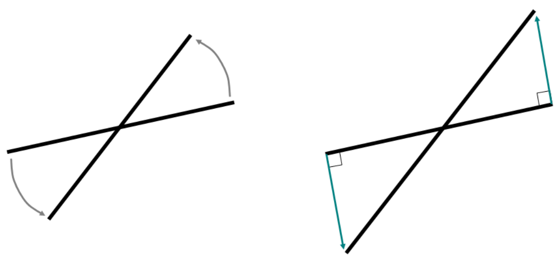


Fig. 7. Normal rotation (left) Linearized rotation (right). The linearized Green strain is invariant to linear rotations allowing unwanted stretching and shearing.

The null space of M can be removed without any artifacts by first fixing a point $\vec{X}_1$ as the origin to remove the translation, i.e. set

$$\vec{u}_1' = 0.$$

Subsequently, we can align $\vec{x}_1 - \vec{x}_2$ with $\vec{X}_1 - \vec{X}_2$ to remove one degree of rotation. Let

$$Proj(\vec{u}, \hat{n}) = (\vec{u} \cdot \hat{n})\hat{n}.$$

Then in terms of displacements, we set

$$\vec{u}_2' = Proj\left(\vec{u}_2, \frac{\vec{X}_2 - \vec{X}_1}{\|\vec{X}_2 - \vec{X}_1\|}\right).$$

Finally, for tetrahedra, we align a third point $\vec{x}_3$ to be in the plane passing through $\vec{X}_1$, $\vec{X}_2$, $\vec{X}_3$ to remove the remaining freedom of rotation, i.e.

$$\vec{u}_3' = \vec{u}_3 - Proj\left(\vec{u}_3, \frac{(\vec{X}_2 - \vec{X}_1) \times (\vec{X}_3 - \vec{X}_1)}{\|(\vec{X}_2 - \vec{X}_1) \times (\vec{X}_3 - \vec{X}_1)\|}\right).$$

Note that the null space gives extraneous translation and linearized rotation, which is dependent only on how we have aligned our reference frame. Hence, it does not effect the stress field computed from the displacements. In comparison, we note that [5] anchors a large number of points in world space, thus enforcing arbitrary boundary conditions that break momentum conservation. Whereas any scheme that removes the null space generates a non-singular M resulting in a unique solution, the physically correct solution is the one that removes the linearized rotations and translations.

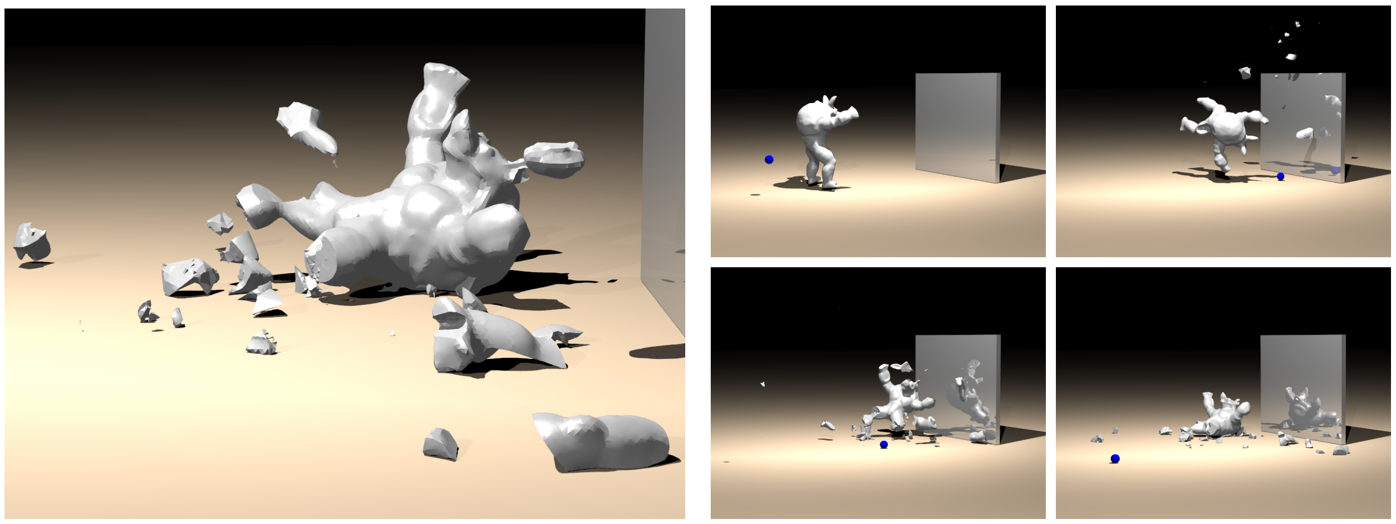


Fig. 8. A volumetric armadillo mesh with 110k tetrahedra is impacted with a high speed projectile. Secondary fracture occurs when the armadillo takes flight and collides with the rigid wall.

VI. Time Averaged Stress

While using stress as the criteria for fracture, plasticity, and damage is a common practice, it incorrectly depends on the size of the time step during collision events. Larger time steps lead to larger overlap between bodies, and thus larger penalty forces during collision. These larger forces create higher stress, which is more likely to cause fracture. The difficulty stems from computing stress and fracture criteria from static snapshots of the forces, which are dependent on the size of the time step for collision events. Instead, we take a time averaged view of the collision based forces, interpreting them via impulses as in [8]. Impulses more accurately measure the strength of a collision event, since they completely resolve the collision as opposed to penalty forces which work over a series of time steps. Figures (9) and (10) illustrates how a snapshot of the force can be arbitrarily high causing spurious fracturing, while the impulse which gives the area under the curve remains constant.

Our rigid-body collision and contact response is impulse based, and we accumulate all such impulses and the points at which they are applied during the rigid body simulation. These impulses are barycentrically distributed to the nodes of the tetrahedra (or triangles) and used in our quasistatic analysis.

Displacements are determined from external forces via $M\vec{u} = -f_{ext}$. This equation can be integrated in time to obtain

$$\int_t^{t+\Delta t} M\vec{u}dt = -\int_t^{t+\Delta t} \vec{f}_{ext} = -\vec{J}_{ext}$$

where $\vec{J}_{ext}$ are the impulses applied to the nodes. Because M is constant, we can solve for the average displacement over the time step as

$$\int_t^{t+\Delta t} \vec{u}dt = -M^{-1}\vec{J}_{ext}.$$

Since stress is a homogeneous function of the displacements, we can simply evaluate stress at the time averaged displacements to obtain a time averaged stress,

$$\sigma|_{\vec{u}=\int_t^{t+\triangle t}\vec{u}dt} = (\partial\sigma(\vec{u})/\partial\vec{u})\int_t^{t+\triangle t}\vec{u}dt =$$
$$\int_t^{t+\triangle t}(\partial\sigma(\vec{u})/\partial\vec{u})\vec{u}dt = \int_t^{t+\triangle t}\sigma(\vec{u})dt$$

which we will use in our fracture criteria. This alleviates the problem of forces, stresses and fracture criteria depending on the size of the time step and subsequent overlap between colliding bodies.

VII. Fracture

We adopt the virtual node algorithm of [10]. Rather than remeshing the computational domain to align with

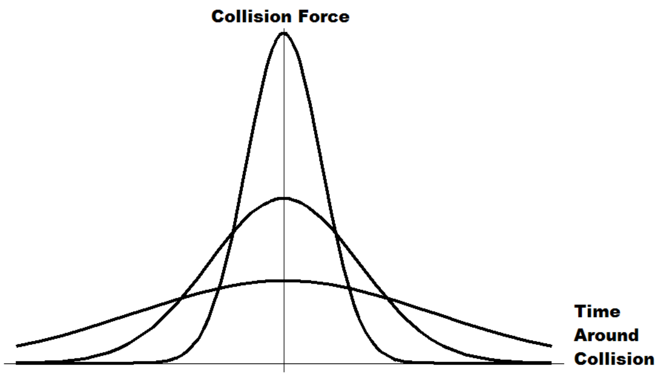


Fig. 9. The collision force is dependent on the size of the time step, while impulse (the area under the curve) is independent of the time step.

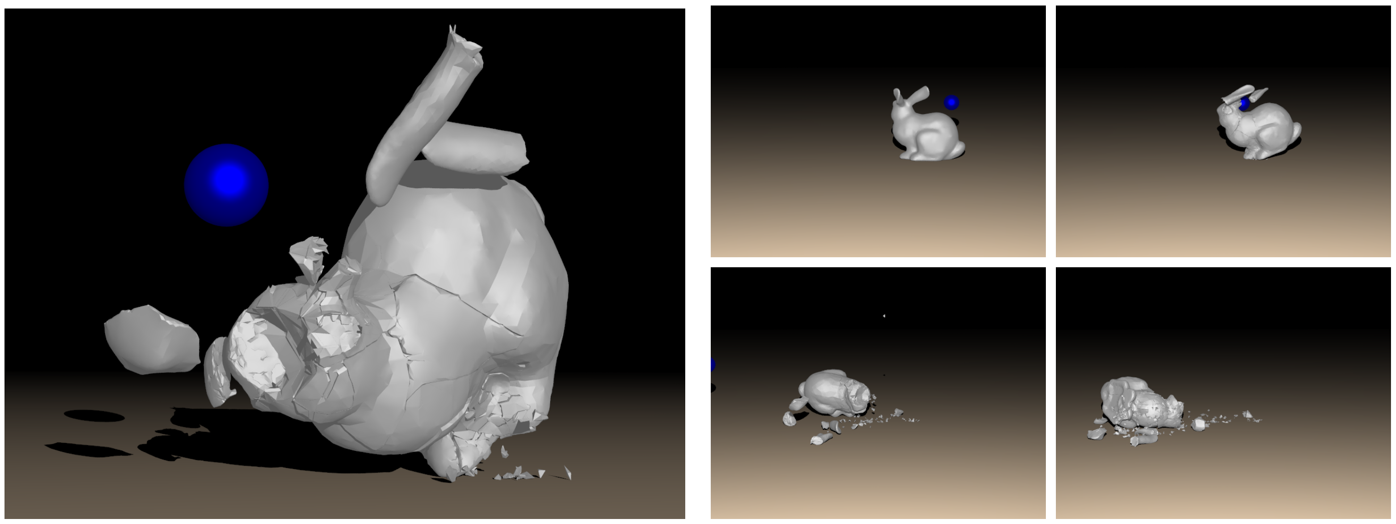


Fig. 11. A bunny mesh with 100k tetrahedra is impacted at high speed. The head is fractured into pieces and fine cracks propagate through the body.

the fracture path, the virtual node algorithm represents the geometry of the fracture surface on element interiors and duplicates mesh elements to create topological changes. As described in [10], duplication occurs one node at a time with a virtual copy of the node created for each distinct scoop carved out of its one ring by the crack surface. This copy is then associated with a new element that represents the piece of the one ring that the original node is now separated from. The popular XFEM method [47] is a special case of the Virtual Node Algorithm obtained by limiting the virtual node algorithm to a single crack per element. In fact [48] cites the virtual node algorithm as ”evolved” from XFEM.

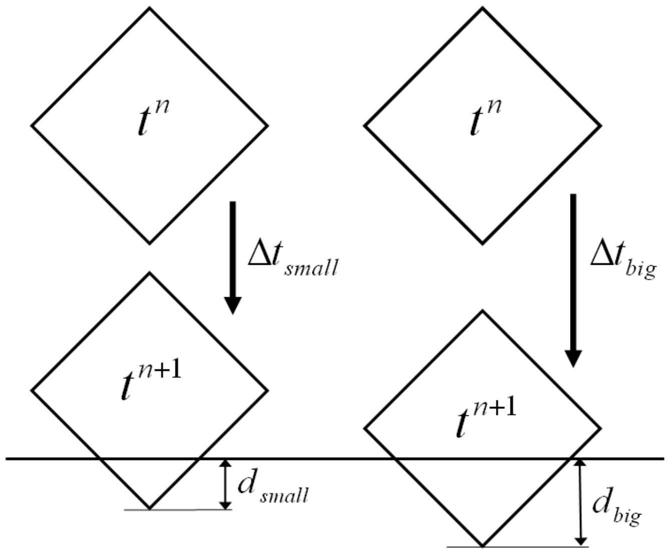


Fig. 10. A smaller time step results in a smaller amount of overlap and thus incurs a smaller penalty force upon penetration (left), while a larger time step results in a larger amount of overlap, incurring a large penalty force and arbitrary fracturing (right).

When an object is topologically separated into two or more pieces, we first calculate the mass and inertia tensor of each piece. This is used to compute the position and orientation. As in [5], to conserve momentum, each rigid body is assigned the velocity that its center of mass had in the non-fractured parent rigid body, i.e. $\vec{v}_i = \vec{v} + \vec{\omega} \times \vec{r}_i$ where r_i points from the center of mass of the parent to the center of mass of child i. Similarly, angular momentum is conserved by setting $\vec{\omega}_i = \vec{\omega}$.

After fracture, we need to compute a new implicit surface for each child rigid body. To make this more efficient, we precompute and store an axis aligned bounding box for each surface triangle in material space. Then for each implicit surface grid node that lies inside this bounding box, we compute the distance to this triangle and store it in the grid node only if it is closer than all other distances computed for that grid node thus far. All this information is precomputed and stored before the simulation begins. This cached structure is further augmented with information from all newly generated triangles during fracture events as the simulation progresses. This gives significant saving when generating these new level sets on-the-fly. The implicit surface is created by marking all edges in these bounding boxes that intersect triangles, flood filling (similar to [5]), finding interior regions, initializing signed distance in the bounding boxes, and fast marching to fill in as large of a band as desired, e.g. see [49].

As increasing numbers of extremely small shards are created it becomes too computationally expensive to generate signed distance functions and too memory intensive to store them. If a shard is too small or if too many have been generated, we simply treat the triangulated boundary of the shard as if it were a thin shell rigid body and process its collisions as outlined in section III.

A. Fracture Criteria

Our fracture criteria is primarily based on our modified Rankine condition. The Rankine condition states that fracture occurs when the principal components of the element stress are above a certain threshold, and that the fracture surface is locally planar as defined by the corresponding principal direction. In our modified version, we use the time averaged stress instead of the instantaneous stress, because any instantaneous stress can be arbitrarily large due to an arbitrarily large penalty force from an arbitrarily large overlap during collision handling. Moreover, to reduce any possible mesh aliasing, we also perform one iteration of volume weighted averaging of the time-averaged stress.

In real material, fracture is also typically initiated along weakness in the material. Many authors in the mechanical engineering community model material weaknesses with grain boundaries in conjunction with the XFEM method, see e.g. [50], [51]. We also incorporate this option for artistic control. In particular, we include a randomly seeded model for surfaces of weakness in the material and bias fracturing along these surfaces. This is accomplished by randomly seeding a number of points, defining an energy based on the weighted distance from each point, and assigning tetrahedral nodes to regions based on minimum energy. In the end, these regions are defined using level sets to smoothly determine a directional bias for the crack formation. For example, if there are two regions with seeds at $\mathbf{p}_i$ and $\mathbf{p}_j$. The distance energy functions of seed points can be defined as $E_i(\mathbf{x}) = k_i \cdot distancefrom(\mathbf{p}_i)$. Hence, the levelset surface ϕ_{ij} that will be used to cut the object for region i is

$$\phi_{ij} = E_i - (E_i + E_j)/2 = (E_i - E_j)/2,$$

and for region j is

$$\phi_{ji} = E_j - (E_i + E_j)/2 = (E_j - E_i)/2$$

where ϕ_{ij} and ϕ_{ji} represent an identical cutting plane with inverted volume representations of inside and outside.

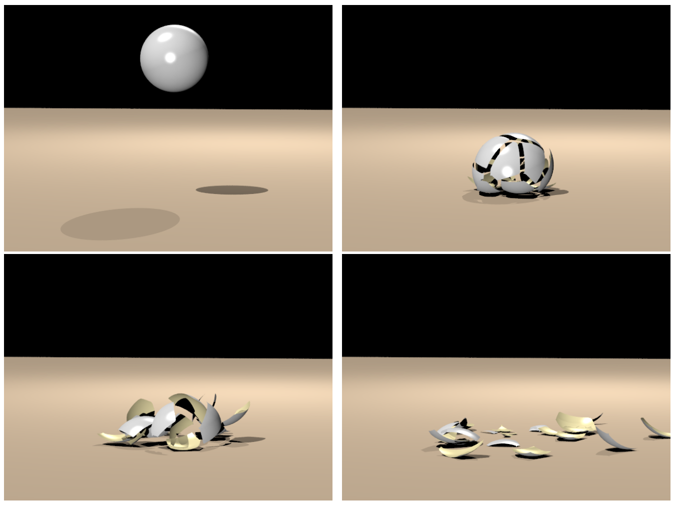


Fig. 12. A single ornament falls and shatters on impact. The fracture patterns are influenced by grain boundaries in the material used to represent imperfections.

B. Plasticity and Denting

While brittle material does not bend, we added a simple denting model for plastic behavior that can be tuned for increased artistic effect. Our approach uses the time averaged displacements determined during the quasistatic stress analysis to give a direction of plastic flow. Using these directions, we simply deform the mesh with an Euler step

$$\vec{x} += \alpha \int_t^{t+\triangle t} \vec{u} dt$$

where α is a scaling constant. This parameter can be adjusted to account for the degree of pliability in the material. Note that our denting model is based on linear amplification of the "time-averaged linear displacements", whereas [5] uses non-linear displacements that contain large $O(1)$ time step dependent errors from their penalty based collision system.

VIII. Examples

We generated all volumetric and surface meshes with the meshing algorithm of [45]. Figures 2 and 6 illustrate our ability to generate and simulate both aesthetically pleasing fracture and physically realistic small shards. The large volumetric examples in figures 8 and 11 demonstrate the rich dynamic fracture achievable when making use of a fast rigid body solver. The shell examples in figures 4, 12 and 1 show our ability to extend the technique to thin shell rigid bodies by exploiting the accuracy and efficiency of our exact triangle collisions algorithm. Finally, we have observed extremely good performance with our algorithm. For example, simulating the fracturing of 100 ornaments into over 2000 pieces consisting of over a million triangles took only 40 seconds per frame on a 3.2 Ghz Pentium 4. Similarly, large volumetric examples such as the bunny and the armadillo took a few seconds per frame prior to fracture and about a minute per frame afterwards.

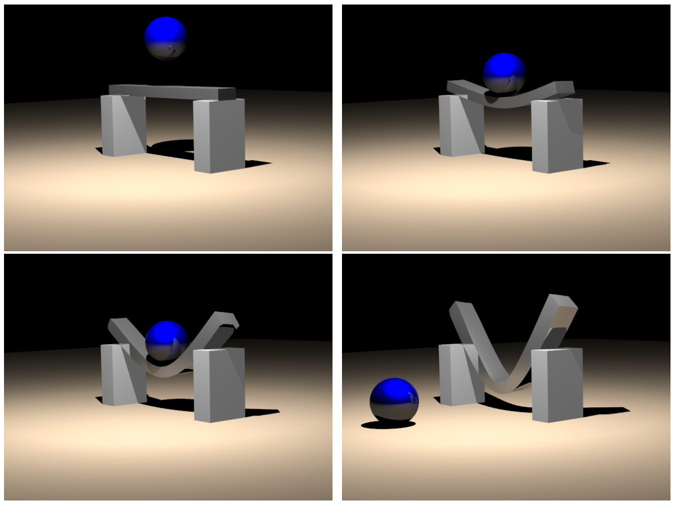


Fig. 13. Plastic bending of a metallic rod during collision. The time averaged displacements from our quasistatic fracture analysis provide an approximation to plastic flow and are used to determine the damaged configuration.

IX. Conclusion and Future Work

We proposed a novel algorithm for efficient and realistic fracture of extremely stiff and brittle materials. Objects were treated as rigid bodies to avoid the stringent time step restrictions of explicit methods and the computational cost of solving nonlinear systems of equations for implicit methods. Rigid body assumptions were also used for collisions and contact, and an extension to extremely thin objects was presented to efficiently manage small shards and thin shells of material. To create physically realistic fracture dynamics, each object was endowed with a linearly elastic constitutive model and quasistatic analysis was used to determine stress distributions. We presented an impulse and time averaged displacement approach to determining stress to ensure time step independent fracturing. We also included a plasticity model for damage and denting to increase the richness of fracture events. However, while the bending model is fast and simple, it causes slight volume increases typical of linear elasticity. In the future, we will explore the addition of non-linear effects to our model for bending, although it matters less for fracture and increases the cost of the method.

X. Acknowledgements

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, ONR N00014-02-1-0720, ONR N00014-05-1-0479 (for a computing cluster), ARO DAAD19-03-1-0331, NIH U54-GM072970, NSF IIS-0326388 and NSF ITR-0205671. Z.B. was partially supported by a NSF Graduate Research Fellowship. J.H. was partially supported by the IT Scholarship Program supervised by the IITA (Institute for Information Technology Advancement) & MIC (Ministry of Information and Communication), Republic of Korea. J.T. was supported by a NSF Mathematical Sciences Postdoctoral Research Fellowship.

References

[1] J. O'Brien and J. Hodgins, "Graphical modeling and animation of brittle fracture," in *Proc. of SIGGRAPH 1999*, 1999, pp. 137–146.

[2] D. Baraff and A. Witkin, "Large steps in cloth simulation," in *Proc. SIGGRAPH 98*, 1998, pp. 1–12.

[3] E. Grinspun, A. Hirani, M. Desbrun, and P. Schröder, "Discrete shells," in *Proc. of the 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2003, pp. 62–67.

[4] J. Teran, E. Sifakis, G. Irving, and R. Fedkiw, "Robust quasistatic finite elements and flesh simulation," *Proc. of the 2005 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2005.

[5] M. Müller, L. McMillan, J. Dorsey, and R. Jagnow, "Real-time simulation of deformation and fracture of stiff materials," in *Comput. Anim. and Sim. '01*, ser. Proc. Eurographics Wrkshp. Eurographics Assoc., 2001, pp. 99–111.

[6] M. Pauly, D. Pai, and L. Guibas, "Quasi-rigid objects in contact," in *Proc. of the 2004 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2004.

[7] D. Terzopoulos and A. Witkin, "Physically based models with rigid and deformable components," in *Graph. Interface*, 1988, pp. 146–154.

[8] E. Guendelman, R. Bridson, and R. Fedkiw, "Nonconvex rigid bodies with stacking," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 22, no. 3, pp. 871–878, 2003.

[9] D. Kaufman, T. Edmunds, and D. Pai, "Fast frictional dynamics for rigid bodies," *ACM Trans. Graph. (SIGGRAPH Proc.)*, pp. 946–956, 2005.

[10] N. Molino, Z. Bao, and R. Fedkiw, "A virtual node algorithm for changing mesh topology during simulation," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 23, pp. 385–392, 2004.

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

[12] Y. Gingold, A. Secord, J. Han, E. Grinspun, and D. Zorin, "A discrete model for inelastic deformations of thin shells," in *Poster, Eurographics/ACM SIGGRAPH Symp. on Comput. Anima.*, 2005.

[13] M. Wicke, D. Steinemann, and M. Gross, "Efficient animation of point-sampled thin shells," in *Proc. of Eurographics*, vol. 24, no. 3, 2005.

[14] M. Pauly, R. Keiser, B. Adams, P. Dutré, M. Gross, and L. Guibas, "Meshless animation of fracturing solids," *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 24, no. 3, pp. 957–964, 2005.

[15] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer, "Elastically deformable models," *Comput. Graph. (Proc. SIGGRAPH 87)*, vol. 21, no. 4, pp. 205–214, 1987.

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

[17] D. Terzopoulos and K. Fleischer, "Deformable models," *The Vis. Comput.*, no. 4, pp. 306–331, 1988.

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

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

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

[21] J. Teran, E. Sifakis, S. Salinas-Blemker, V. Ng-Thow-Hing, C. Lau, and R. Fedkiw, “Creating and simulating skeletal muscle from the visible human data set,” *IEEE Trans. on Vis. and Comput. Graph.*, vol. 11, no. 3, pp. 317–328, 2005.

[22] G. Debunne, M. Desbrun, M. Cani, and A. Barr, “Dynamic real-time deformations using space & time adaptive sampling,” in *Proc. SIGGRAPH 2001*, vol. 20, 2001, pp. 31–36.

[23] E. Grinspun, P. Krysl, and P. Schröder, “CHARMS: A simple framework for adaptive simulation,” *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 21, pp. 281–290, 2002.

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

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

[26] O. Etzmuss, M. Keckeisen, and W. Strasser, “A fast finite element solution for cloth modelling,” in *Pacific Graph.*, 2003, pp. 244–251.

[27] M. Müller and M. Gross, “Interactive virtual materials,” in *Graph. Interface*, May 2004, pp. 239–246.

[28] M. G. Choi and H.-S. Ko, “Modal warping: Realtime simulation of large rotational deformation and manipulation,” *IEEE Trans. Viz. Comput. Graph.*, vol. 11, pp. 91–101, 2005.

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

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

[31] J. Barbic and D. James, “Real-time subspace integration of st. venant-kirchoff deformable models,” *ACM Trans. Graph. (SIGGRAPH Proc.)*, pp. 982–990, 2005.

[32] M. Müller, B. Heidelberger, M. Teschner, and M. Gross, “Meshless deformations based on shape matching,” *ACM Trans. Graph. (SIGGRAPH Proc.)*, pp. 471–478, 2005.

[33] N. Bell, Y. Yu, and P. J. Mucha, “Particle-based simulation of granular materials,” in *SCA ’05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. New York, NY, USA: ACM Press, 2005, pp. 77–86.

[34] A. Norton, G. Turk, B. Bacon, J. Gerth, and P. Sweeney, “Animation of fracture by physical modeling,” *Vis. Comput.*, vol. 7, no. 4, pp. 210–219, 1991.

[35] K. Hirota, Y. Tanoue, and T. Kaneko, “Generation of crack patterns with a physical model,” *The Vis. Comput.*, vol. 14, pp. 126–187, 1998.

[36] O. Mazarak, C. Martins, and J. Amanatides, “Animating exploding objects,” in *Proc. of Graph. Interface 1999*, 1999, pp. 211–218.

[37] J. Smith, A. Witkin, and D. Baraff, “Fast and controllable simulation of the shattering of brittle objects,” in *Comput. Graph. Forum*, D. Duke and R. Scopigno, Eds. Blackwell Publishing, 2001, vol. 20(2), pp. 81–91.

[38] D. Bielser, P. Glardon, M. Teschner, and M. Gross, “A state machine for real-time cutting of tetrahedral meshes,” in *Pacific Graph.*, 2003, pp. 377–386.

[39] D. Bielser and M. Gross, “Interactive simulation of surgical cuts,” in *Pacific Graph.*, 2000, pp. 116–125.

[40] J. O’Brien, A. Bargteil, and J. Hodgins, “Graphical modeling of ductile fracture,” *ACM Trans. Graph. (SIGGRAPH Proc.)*, vol. 21, pp. 291–294, 2002.

[41] G. Yngve, J. O’Brien, and J. Hodgins, “Animating explosions,” in *Proc. SIGGRAPH 2000*, vol. 19, 2000, pp. 29–36.

[42] M. Neff and E. Fiume, “A visual model for blast waves and fracture,” in *Proc. of Graph. Interface 1999*, 1999, pp. 193–202.

[43] M. Müller, M. Teschner, and M. Gross, “Physically-based simulation of objects represented by surface meshes,” in *Proc. Comput. Graph. Int.*, June 2004, pp. 156–165.

[44] B. Mirtich, “Fast and accurate computation of polyhedral mass properties,” *J. Graph. Tools*, vol. 1, no. 2, pp. 31–50, 1996.

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

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

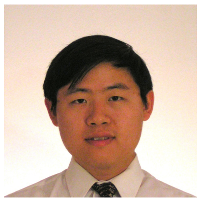
[47] C. Daux, N. Moes, J. Dolbow, N. Sukumar, and T. Belytschko, “Arbitrary branched and intersecting cracks with the extended finite element method,” *Int. J. Num. Meth. Eng.*, vol. 48, pp. 1741–1760, 2000.

[48] P. Areias and T. Belytschko, “Analysis of three-dimensional crack initiation and propagation using the extended finite element method,” *Int. J. Num. Meth. Eng.*, vol. 63, pp. 760–788, 2005.

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

[50] A. Simone, C. A. Duarte, and E. V. der Giessen, “A generalized finite element method for polycrystals with discontinuous grain boundaries,” *Int. J. Num. Meth. Eng.*, vol. 69, p. in press, 2006.

[51] N. Sukumar, D. J. Srolovitz, T. J. Baker, and J.-H. Prévost, “Brittle fracture in polycrystalline microstructures with the extended finite element method,” *Int. J. Num. Meth. Eng.*, vol. 56, pp. 2015–2037, 2003.



Zhaosheng Bao received his B.S. degree from Caltech in 2002 and his Ph.D. degree in computer science from Stanford University in 2006. He was a NSF Graduate Fellow, focusing on fracture animation and simulating topology change in deformable bodies without remeshing. Prior to Stanford, he has also worked as a Research Fellow at the Caltech Graphics Lab, the Jet Propulsion Lab (NASA), the MIT Mathematics department, and the CERN/HP center for High Energy Physics. He was also a lead programmer at ehongkong.com.

Jeong-Mo Hong received his BS degree in 2000 and the MS degree in 2002 in Mechanical Engineering from Korea Advanced Institute of Science and Technology (KAIST) and his PhD degree in Computer Science from Korea University in 2005. Dr. Hong is currently a Postdoctoral Scholar at Stanford University where he is working with Professor Ronald Fedkiw. His research is focused on the visual simulation techniques for computer graphics applications.

Joseph Teran received his Ph.D. in Scientific Computing/Computational Mathematics from Stanford University in 2005 where he was a NSF Graduate Research Fellow and a member of Professor Ronald Fedkiw's research group. His work there was focused on computer graphics and biomechanics applications of muscle and soft tissue simulation. Dr. Teran is currently a NSF Mathematical Sciences Postdoctoral Fellow at the Courant Institute of NYU where he is working with Professor Charlie Peskin on solid/fluid coupling and computational biomechanics. Dr. Teran has also worked as a consultant with Honda Research Institute, Sony Imageworks and Simquest Inc.

Ron Fedkiw received his Ph.D. in Mathematics from UCLA in 1996 and did postdoctoral studies both at UCLA in Mathematics and at Caltech in Aeronautics before joining the Stanford Computer Science Department. He was awarded the National Academy of Science Award for Initiatives in Research, a Packard Foundation Fellowship, a Presidential Early Career Award for Scientists and Engineers (PECASE), a Sloan Research Fellowship, the ACM Siggraph Significant New Researcher Award, an Office of Naval Research Young Investigator Program Award (ONR YIP), a Robert N. Noyce Family Faculty Scholarship, two distinguished teaching awards, etc. Currently he is on the editorial board of the Journal of Scientific Computing, IEEE Transactions on Visualization and Computer Graphics, and Communications in Mathematical Sciences, and he participates in the reviewing process of a number of journals and funding agencies. He has published over 60 research papers in computational physics, computer graphics and vision, as well as a book on level set methods. For the past five years, he has been a consultant with Industrial Light + Magic. He received screen credits for his work on ”Terminator 3: Rise of the Machines”, ”Star Wars: Episode III - Revenge of the Sith”, and ”Poseidon”.