# Real-Time Simulation of Deformation and Fracture of Stiff Materials

Matthias Müller    Leonard McMillan    Julie Dorsey    Robert Jagnow
Laboratory for Computer Science, Massachusetts Institute of Technology

**Abstract.** Existing techniques for real-time simulation of object deformation are well suited for animating soft materials like human tissue or two-dimensional systems such as cloth. However, simulation of deformation in malleable materials and fracture in brittle materials has only been done offline because the underlying equations of motion are numerically stiff, requiring many small steps in explicit integration schemes. In contrast, the better-behaved implicit integration techniques are more expensive per time step, particularly for volumetric meshes. We present a stable hybrid method for simulating deformation and fracture of materials in real-time. In our system, the effects of impact forces are computed only at discrete collision events. At these impacts, we treat objects as if they are anchored and compute their static equilibrium response using the Finite Element technique. Static analysis is not time-step bound and its stability is independent of the stiffness of the equations. The resulting deformations, or possible fractures, are computed based on internal stress tensors. Between collisions, disconnected objects are treated as rigid bodies. The simulator is demonstrated as part of a system that provides the user with physically-based tools to interactively manipulate 3D models.

## 1 Introduction

Modeling and simulation of deformable objects has a long history in material sciences and engineering. In computer graphics, deformable objects have been studied for nearly two decades [6], but, with very different objectives. In graphics applications, the primary concern is usually the computational efficiency of generating plausible behaviors, rather than the accurate prediction of exact results. As long as the simulation looks realistic, simplifications are deemed acceptable.

Deformable models have been used across a wide range of computer graphics applications, including the animation of cloth, facial expressions, and general non-rigid models. In these cases, the simulation is typically performed off-line because simulation time is significantly slower than wall clock time.

Another application for deformable models is in real-time systems, such as surgical training and virtual sculpting, where users interactively modify deformable models. In real-time systems, the speed of the simulator and its stability are the two major concerns. One approach toward achieving both performance and robustness is to use simplified physical models, such as mass-spring models [6]. However, it is difficult to express important material properties with these approaches, such as the stress-strain relationship. Alternatively, the computational cost of continuum methods are considerably more expensive but they allow for the modeling of volume conservation and yield stress information that is useful for determining fracture positions and orientations. Continuum models have mainly been used in off-line simulators. Real-time performance has only been achieved in the simulation of soft materials such as human tissue or in the simulation of cloth represented by a 2D mesh.

In this paper, we describe techniques for simulating the deformation of malleable materials, such as soft metals and plastics, and the fracture of brittle materials, such as stone or glass. Our approach employs continuum methods and operates in real-time. We have integrated these techniques into a physically-based animation system for objects represented by volumetric tetrahedral meshes. Simulating the dynamics of such materials is computationally expensive because explicit integration of the equation of motion is stable only for small time steps. The range of possible time steps is bound by the largest natural frequency of the system. In contrast, implicit integration is stable independent of the size of the time step, but at every step, a linear system of equations has to be solved, which makes it computationally much more expensive per time step. Implicit integration as been used in real-time simulation, but only for 2D meshes, such as cloth [1].

We solve these problems by exploiting the fact that the transient behavior of stiff materials can be neglected in a real-time simulation without significant loss of realism. The natural frequencies of stiff materials tend to be higher than the frame rate of the simulator, and these vibration modes are quickly damped, at least visually, by the object's mass. This makes simulation of these high frequency vibrations dispensable. As long as an object is anchored, we only compute its static equilibrium response to forces. No time steps are involved in the process of finding the object's final steady state. By using this approach, we still have access to all the stress information needed to deform the model and/or simulate its fracture. However, static analysis cannot be used to compute the trajectories of disconnected pieces. Because we neglect internal vibrations, we treat disconnected pieces as rigid bodies between impacts. Their trajectories are computed using rigid body dynamics. When a collision occurs, we compute the effect of the impact force using a static analysis. The body is fractured according to its internal stresses, and new bodies are generated if the fracture process causes fragments to become disconnected.

## 1.1 Related Work

To improve the numerical stability of the simulation of stiff materials Terzopoulos *et al.*[11] proposed a hybrid model that breaks a deformable object into a rigid and a deformable component. The rigid reference body captures the rigid-body motion, while a discretized displacement function gives the location of mesh nodes relative to their position within the rigid body. This method is intended to improve the numerical condition of the underlying equations of motion, however, it does not significantly decrease the computational cost of the simulation. Our approach is similar, but it is hybrid in time. We neglect the displacements from the rigid reference frame between collision events, which makes the simulation both stable and fast. However, when objects collide, we compute the displacements, stresses and fracture bases on a continuous model.

O'Brien *et al.* [7], described a technique for simulating brittle fracture of stiff materials. They discretized the continuum mechanics equations using the Finite Element method based on constant strain tetrahedra. Their use of an explicit integration scheme restricts the time step of the simulation to very small values that are not suitable for real-time animation. Another problem is the crack-tip propagation or growth rate. Because the cracks can only grow one tetrahedron per time step, many iterations are required to break an object in two or more pieces. Our approach is similar in that we use the same continuous model and the constant-strain tetrahedron approximation to compute displacements and stress tensors. However, we solve for static equilibrium configurations rather than integrating the general equation of motion, and we do this only at collision events. We use the orientation of stress tensors to compute fracture planes and cracks, thus making the rate and size of crack propagation independent of the tetrahedral mesh's granularity.

Real-time performance in simulating deformation based on continuous models has been achieved for soft materials such as human tissue for use in virtual surgical training systems. Zhuang [13] uses the Finite Element method to simulate global deformation of human tissue in real-time. However, his explicit integration scheme is appropriate only for soft materials and not suitable for cloth — which is stiff in certain directions — or other stiff materials like plastic or stone.

Baraff *et al.*[1] describe a technique for simulating cloth using an implicit integration scheme. The implicit integration method can take large time steps without loss of stability. However, for every time step, a system of linear equations has to be solved. Cloth is represented as a 2D mesh of triangles. The method is not as well suited for 3D meshes of tetrahedra. First, the number of vertices is substantially larger in a volumetric model, and second, the linear system is not as "banded" as in the 2D case, which makes implicit integration computationally expensive for 3D objects.

Recently, Smith *et al.*[9] have proposed a novel approach for simulating brittle fracture of stiff materials in real-time. They represent objects as a set of point masses connected by distance-preserving linear constraints. The forces exerted by these constraints during impact are computed using Lagrange multipliers. In contrast to our approach, these rigid constraints do not allow for computing object deformations caused by collision forces, nor do they yield strain orientation information that we need for our fast crack propagation procedure.

### 1.2 Overview

In the next section, we describe the continuous model that we use. We discretize it using the Finite Element method based on constant strain tetrahedra. First we provide an overview of standard techniques to compute static elastic and plastic responses. Then we introduce our new hybrid algorithm to simulate the dynamics of freely moving objects. Then we show how to accelerate the core procedures of the Finite Element method. Last, we present a collection of our results.

## 2 Modeling Deformation

Our virtual animation system provides tools for manipulating objects. These objects are represented by 3-dimensional tetrahedral meshes. A tool generates a local force field. The shape of that force field depends on the type of tool as well as on the direction and intensity of its application. The task of the physical simulator is to compute the deformation of the object and the fracturing process based on the applied force field.

A variety of models have been used to simulate the behavior of deformable objects. Mass-spring models are simple and fast to compute. However, models that treat objects as a continuum yield a range of important additional information, not to mention results that are more accurate. The deformations of objects in a continuous model are described by a set of partial differential equations. For realistic objects, these equations cannot be solved analytically. The Finite Element method is a standard technique to solve partial differential equations [2]. Here, the object is subdivided into elements of finite size. Over an element, the continuous deformation field is interpolated from deformation values at the nodes. By connecting elements, the deformation field is interpolated over the entire object in piecewise continuous fashion. Instead of solving for a continuous vector field, deformations at discrete points or nodes in the objects have to be computed, and the differential equations at these nodes are treated as set of simultaneous algebraic equations.

### 2.1 Continuous Model

In one dimension, Hooke's law of elasticity can be stated as follows:

$$\sigma = \frac{\Delta F_n}{\Delta A} = E\frac{\Delta l}{l} = E\varepsilon. \tag{1}$$

The scalar stress $\sigma$ measures the force $\Delta F_n$ applied perpendicular to the surface $\Delta A$. This force causes a deformation (strain) $\varepsilon$ of the object measured by the change in length perpendicular to $\Delta A$ with respect to the original length of the object. The scalar elasticity Modulus $E$ relates the strain $\varepsilon$ to the stress $\sigma$.

In three dimensions, forces, orientations of surfaces, and node displacements can be represented as 3-dimensional vectors, and the quantities that relate them, namely $\sigma$ and $\varepsilon$ can be expressed as 3 by 3 matrices. The derivation of these tensors can be found in continuum mechanics textbooks [3]. In this paper, we focus primarily on how these quantities can be computed efficiently. The following equations are very similar to those presented by O'Brien *et al.* [7], although we have chosen to use matrix notation for reasons of compactness and ease of manipulation. Matrix notation also exposes various symmetries that we will later take advantage of to speed up the computation of forces.

Let $\mathbf{u} = [u_1, u_2, u_3]^T$ be the spatial coordinates of an undeformed object point. The deformation of the object can be described by a function $\mathbf{p}(\mathbf{u}) = [p_1, p_2, p_3]^T$, which maps locations in the undeformed coordinate frame to locations in world coordinates. This function must be differentiable within connected pieces of the object. In three dimensions, there are several ways to measure deformation. One approach is to use Green's strain tensor, which is invariant with respect to rigid body transformations applied to $\mathbf{p}$, and vanishes when the material is not deformed. It is accurate for arbitrary deformations. However, the fact that it is non-linear causes some difficulties that we will treat later. Green's 3 by 3 symmetric tensor reads:

$$\varepsilon = \mathbf{J}_u(\mathbf{p})\mathbf{J}_u^T(\mathbf{p}) - \mathbf{I}, \tag{2}$$

where $\mathbf{J}_u(\mathbf{p})$ is the Jacobian of the vector function $\mathbf{p}$ with respect to the vector $\mathbf{u}$.

Hooke's law relates stress, $\sigma$, to the strain, $\varepsilon$. For isotropic materials, this relation can be expressed using only two constants $\mu$ and $\lambda$, which are the Lamé constants of the material:

$$\sigma = 2\mu\varepsilon + \lambda\mathrm{Trace}(\varepsilon)\mathbf{I} \tag{3}$$

Both the strain and stress tensors are symmetric and functions of the material coordinates $\mathbf{u}$. They are used to compute the elastic potential density, $\eta$, as

$$\eta = \frac{1}{2}\mathrm{Trace}(\sigma\varepsilon). \tag{4}$$

The total elastic potential is obtained by integrating $\eta$ over the volume of the body. According to the principles of energy conservation, the internal work (elastic potential) has to be equal to the external work done by the external forces. Thus, given an external force field, the deformation function $\mathbf{p}$ can be computed as the solution to a partial differential equation.

## 2.2 Finite Element Formulation

The Finite Element method approximates the deformation function $\mathbf{p}$ as piecewise smooth between discrete elements. The elements can be of arbitrary shape as long as they share nodes and faces with adjacent elements and cover the region of interest. We use tetrahedral meshes because they are simple, flexible and computationally inexpensive. Within a tetrahedron, a linear approximation of $\mathbf{p}$ is used. Such linear deformations yields constant strain and stress tensors within each element. Therefore, these quantities can be moved outside of any integration over an element's volume. Like O'Brien *et al.* [7], we assume constant strain tetrahedra and use barycentric coordinates for interpolating within them. We will restate these formulas and later show how to compute them efficiently as well as show how to compute the static equilibrium using a non-linear strain tensor.

Let $\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3, \mathbf{m}_4$ be the coordinates of the four nodes of a tetrahedron in the undeformed material coordinate frame, and let $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4$ be their deformed world coordinates. First we need the linear continuous deformation function $\mathbf{p}(\mathbf{u})$ for this tetrahedron, which maps $\mathbf{m}_i$ to its corresponding $\mathbf{x}_i$. Let $\mathbf{b} = [b_1, b_2, b_3, b_4]^T$ be barycentric coordinates defined in terms of the element's nodal positions in the undeformed coordinate frame.

$$\begin{bmatrix} \mathbf{u} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{m}_1 & \mathbf{m}_2 & \mathbf{m}_3 & \mathbf{m}_4 \\ 1 & 1 & 1 & 1 \end{bmatrix} \mathbf{b}. \tag{5}$$

We use these barycentric coordinates $\mathbf{b}$ to identify the interpolated point, $\mathbf{u}$, with its corresponding position in world coordinates, $\mathbf{p}$:

$$\begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_4 \\ 1 & 1 & 1 & 1 \end{bmatrix} \mathbf{b}. \tag{6}$$

These relations can be combined to define a direct mapping

$$\mathbf{p}(\mathbf{u}) = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_4 \\ 1 & 1 & 1 & 1 \end{bmatrix} \beta \begin{bmatrix} \mathbf{u} \\ 1 \end{bmatrix}, \tag{7}$$

where

$$\beta = \begin{bmatrix} \mathbf{m}_1 & \mathbf{m}_2 & \mathbf{m}_3 & \mathbf{m}_4 \\ 1 & 1 & 1 & 1 \end{bmatrix}^{-1}. \tag{8}$$

This defines our linear deformation function $\mathbf{p}$, allowing the computation of the strain tensor, $\varepsilon$, the stress tensor, $\sigma$ and the potential density $\eta$ defined in Eq. 2, Eq. 3 and Eq. 4. These terms turn out to be constant within each element.

The elastic force on the $i$th node, $\mathbf{f}_i$, is defined as the partial derivative with respect to $\mathbf{x}_i$ of the elastic potential density, $\eta$, integrated over an element's volume. Using Eq. 4 and Eq. 7 we get

$$\mathbf{f}_i = \frac{v}{2} \beta \mathbf{G} \sigma \mathbf{G}^{\mathbf{T}} \beta^{\mathbf{T}} \mathbf{x}_i, \tag{9}$$

where $v$ is the element's volume in the undeformed coordinate frame and where

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}. \tag{10}$$

In Section 3 we discuss how to compute these force vectors efficiently by exploring symmetry and other properties of the strain and stress tensors.

In order to compute the static equilibrium, we also need to compute the Jacobian of the internal forces and stresses with respect to the nodal positions $\mathbf{x_i}$. First, we rewrite Eq. 9:

$$[\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3, \mathbf{f}_4]^T = F'_e(\mathbf{x}_1, .., \mathbf{x}_4). \tag{11}$$

Only the deformed coordinates, $\mathbf{x}_i$, of $F'_e$ vary since the undeformed coordinates $\mathbf{m}_i$ are constant during the animation. The index $e$ represents element number $e$ in the mesh. For technical reasons, we expand $F'_e$ to the unprimed function $F_e$ which has the positions of all $N$ nodes in the mesh as input and produces force vectors for all nodes. It ignores positions of nodes that do not belong to element $e$ and produces zero forces for these nodes. Now, the global function $F$ can be computed as a sum of all the $F_e$'s, as forces coming from adjacent tetrahedra can be added at the nodes

$$[\mathbf{f}_1, .., \mathbf{f}_N]^T = F(\mathbf{x}_1, .., \mathbf{x}_N) = \sum_{e=1}^{E} F_e(\mathbf{x}_1, .., \mathbf{x}_N), \tag{12}$$

or simply, $\mathbf{f} = F(\mathbf{x})$, where $\mathbf{f} = [\mathbf{f}_1, .., \mathbf{f}_N]^T$ and $\mathbf{x} = [\mathbf{x}_1, .., \mathbf{x}_N]^T$.

### 2.3 Static Analysis

In a static analysis, we solve for the positions of all nodes ($\mathbf{x}$) such that the internal forces $F(\mathbf{x})$ are in balance with the externally applied forces $\mathbf{f}_{\text{ext}}$

$$F(\mathbf{x}_{\text{eq}}) = \mathbf{f}_{\text{ext}}. \tag{13}$$

To compute the coordinates $\mathbf{x}_{\text{eq}}$, we have to solve a nonlinear system of $3N$ equations. The non-linearity of $F$ is due to the fact that we are using a non-linear strain tensor in Eq. 2. The most common method to solve systems of non-linear algebraic equations is the Newton-Raphson iteration [8]. First we replace $F(\mathbf{x})$ by its first-order Taylor series approximation at $\mathbf{x}_k$:

$$F(\mathbf{x}_k + \Delta\mathbf{x}) = F(\mathbf{x}_k) + J(\mathbf{x}_k)\,\Delta\mathbf{x} + O(\|\Delta\mathbf{x}\|^2) \tag{14}$$

where $J \in \mathbb{R}^{3N\times 3N}$ is the Jacobian of $F$ and $J_{ij} = \frac{\partial F_i}{\partial x_j}$. We can now rewrite Eq. 13 as

$$F(\mathbf{x}_{\text{eq}}) = F(\mathbf{x}_k + \Delta\mathbf{x}) \approx F(\mathbf{x}_k) + J(\mathbf{x}_k)\,\Delta\mathbf{x} = \mathbf{f}_{\text{ext}} \tag{15}$$

or

$$J(\mathbf{x}_k)\,\Delta\mathbf{x} = \mathbf{f}_{\text{ext}} - F(\mathbf{x}_k). \tag{16}$$

Given an estimate of $\mathbf{x}_k$ for $\mathbf{x}_{\text{eq}}$, we first evaluate $J$ at position $\mathbf{x}_k$ and solve this linear system for $\Delta\mathbf{x}$ using the iterative Conjugate Gradients method [8]. Then, $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta\mathbf{x}$ is the next guess for $\mathbf{x}_{\text{eq}}$.

### 2.4 Simulating Plastic Behavior

So far, our analysis has dealt with only perfectly elastic objects. Such objects remain deformed as long as forces are applied. When the forces are removed, such objects resume their original shape. There is another interesting class of materials that exhibit

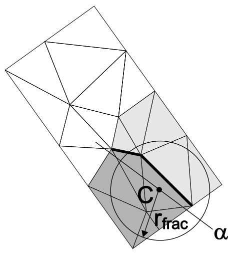

**Fig. 1.** Tetrahedra within radius $r_{frac}$ from the tetrahedron under tensile stress are marked depending on their position with respect to the fracture plane $\alpha$

plastic deformations, such as malleable metals or clay. A perfectly plastic material absorbs the elastic energy. That is, it keeps its deformed shape when the forces are removed.

To simulate the plastic behavior of malleable materials, we use a technique similar to Terzopoulos and Fleischer in [10]. Whenever the deformed coordinates $\mathbf{x}$ deviate too much from the original shape of the object $\mathbf{m}$, we copy the deformed world coordinates to the undeformed coordinates, thereby absorbing the elastic potential energy. Figure 4 (see Appendix) shows a clay teddy bear. The deformations caused by impacts are absorbed and accumulate over time.

## 2.5 Fracture Modeling

In order to simulate fracture at interactive rates we devised a simplified physically-based method to separate tetrahedra. In the process of computing the forces for the static analysis, the stress tensors for all tetrahedra in the mesh have to be evaluated. The stress tensor $\sigma$ is a symmetric 3 by 3 matrix and has thus 3 real eigenvalues. These eigenvalues correspond to the principal stresses, and their eigenvectors to the principal stress directions [2]. A positive eigenvalue indicates tension while a negative value represents compression.

The maximum tensile stress criterion assumes that fracture of a material occurs when the maximum tensile stress exceeds a specific stress threshold, which is a material parameter. For all tetrahedra, we evaluate the largest eigenvalue $d_{max}$ of $\sigma$. If $d_{max}$ is greater than the fracture threshold of the material, we split the tetrahedral mesh along a plane $\alpha$ perpendicular to the eigenvector of $d_{max}$. Most isotropic materials break in this way, since this is how the greatest deformation energy is released [5]. Depending on the size of $d_{max}$ and the material type, we determine a radius $r_{frac}$ of impact. All tetrahedra within distance $r_{frac}$ from the greatest stress tetrahedron, where the crack originates in our model, are marked with a plus or a minus depending on whether their center of mass lie on the positive or negative side of the fracture plane $\alpha$. Then, tetrahedra with opposite signs are disconnected (see Figure 1). We also use the orientation and position of $\alpha$ to split large tetrahedra before the mesh is separated.

O'Brien [7] models fracture by splitting single tetrahedra per time step. A dynamic crack growth simulation over multiple time steps is more accurate than our technique. However, realistic results can only be achieved with very small time steps because cracks within brittle materials propagate at very high speeds (at approximately the speed of sound within the material) [5]. The crack growth rate of our technique is independent of both, the time step and the granularity of the tetrahedral mesh. Both properties are crucial if the time step size of the simulator does not permit the computation of a more accurate crack propagation. Moreover, the fact that cracks in homogeneous isotropic materials tend to be locally planar [5] justifies our simplified approach.

## 2.6 Dynamics

Static analysis can only be performed for supported objects. We therefore anchor our models to a ground plane before forces are applied — just as objects have to be fixed to a workbench before they can be machined. For simulating free floating objects, an anchoring method that captures their motion and dynamics is needed. The standard technique for simulating the dynamics of deformable objects is to integrate Newton's equations of motion using numerical methods like Euler's integration scheme. The equations of motion used in conjunction with the Finite Element method have the following form:

$$M\ddot{\mathbf{x}} + C\dot{\mathbf{x}} + F(\mathbf{x}) = \mathbf{f}_{\text{ext}}, \tag{17}$$

where the coordinates $\mathbf{x}$ are functions of time, $\dot{\mathbf{x}}$ and $\ddot{\mathbf{x}}$ their time derivatives, $M$ is the mass matrix and $C$ the damping matrix [4]. At equilibrium, when $\mathbf{x} = \mathbf{x}_{\text{eq}}$, $\ddot{\mathbf{x}} = \mathbf{0}$ and $\dot{\mathbf{x}} = \mathbf{0}$, Eq. 17 becomes Eq. 13. The dynamic equation defines a coupled system of $3N$ ordinary differential equations.

## 2.7 Hybrid Dynamics

It is possible to simulate a few hundred elements in real-time using implicit integration of Eq. 17 and the fast Jacobian-computation discussed in Section 3. However, for simulating even larger models efficiently, we have devised a hybrid dynamics approach. The key idea is to separate rigid body dynamics from internal effects such as vibration and fracturing. We evaluate elastic forces only during collision events while treating the body as rigid otherwise. This is a reasonable simplification for stiff materials. As discussed previously, the natural frequencies of stiff materials tend to be much higher than rendering frame rates, and these vibrations are quickly damped. Therefore, this approximation has little impact on the visualization. Malleable materials that absorb the deformation energy of collisions can also be modeled using this simplification if we assume that all deformations occur at the instant of contact.

We treat free-floating objects as rigid bodies and compute their dynamic behavior using rigid body dynamics based on explicit Euler integration [12]. Each rigid body has four state variables, its position $\mathbf{x}_{\text{cm}}$, its center of mass velocity $\mathbf{v}_{\text{cm}}$, its rotational orientation $A$, and, its angular velocity $\omega$. In general, these states can be initialized according to specific user inputs or according to simulation objectives. In the case where a new rigid body can also be generated due to a fracture, the state of all elements in the child components are initialized based on the previous state of the parent object. Each child's state variables are initialized as follows:

$$\begin{aligned} \mathbf{v}^0_{\text{cm}} &= \mathbf{v}^p_{\text{cm}} + \omega^p \times (\mathbf{x}_{\text{cm}} - \mathbf{x}^p_{\text{cm}}) \\ A^0 &= \text{Identity} \\ \omega^0 &= \omega^p \end{aligned} \tag{18}$$

where the superscript $p$ indicates a state value of the parent body.

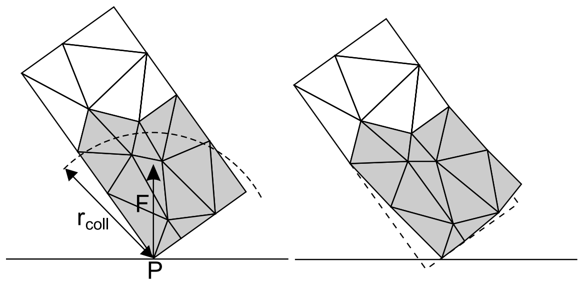

**Fig. 2.** Only tetrahedra within radius $r_{coll}$ from collision point $P$ are deformed. All other tetrahedra are fixed and support the object for a static analysis

## 2.8 Collision Response

Whenever a collision occurs, the effect of the impact force $\mathbf{F}$ on the body is evaluated. Here, our method deviates from a pure rigid body simulation. As a first step, we compute the deformation caused by the collision force using Eq. 13. In rigid body dynamics, the change in impulse $\Delta\mathbf{j}$ is computed rather than a collision force. We evaluate the force as $\mathbf{F} = \Delta\mathbf{j}/\Delta t$, where $\Delta t$ is a simulation parameter representing the duration of the impact. For a static analysis, the body must be anchored. We fix the positions of all tetrahedra that are further away from collision point $P$ than a distance $r_{coll}$ (see Figure 2). The radius $r_{coll}$ is a user-specified simulation parameter that is typically defined as fraction of the size of the rigid body. Anchored tetrahedra model the effect of the body's inertia.

Once the deformed coordinates are determined, the stress tensors and fracture planes can be computed as described in Section 2.5. After the fracture process, the old body is deleted and one or more child bodies are generated, depending on whether the body gets disconnected. These new bodies inherit dynamic properties from their parent body via Eq. 18.

## 2.9 Collision Detection

Performing real-time collision detection between deformable rigid bodies proves challenging for a variety of reasons. Vertex positions within a single rigid body are constantly changed as the body is deformed by collision forces. This dynamic characteristic of the data limits possibilities for precomputing efficient data structures. It also creates a potential for intra-object collisions. Furthermore, the rigid bodies in our system are rarely convex, which limits the use of common closest-feature tracking algorithms. Finally, as new bodies are generated by the fracture algorithm, the faces of the new rigid body are in close proximity to the faces of the parent body. Thus, even though the bodies are not in contact with one another, boundary hierarchy algorithms will likely have to traverse the data to each of its leaf nodes and check each pair of neighboring faces for intersection. Bounding hierarchies are efficient when the rigid bodies are separated, but cumbersome during the fracture process, when bodies tend to be closely aligned along irregularly shaped interfaces.

To determine regions of possible collision, we divide our model space into a regular three-dimensional grid, and then walk through all of the rigid bodies, marking each grid cell with the rigid bodies that lie inside. For cells containing multiple rigid bodies, we look for intersections between the vertices of one body and the tetrahedral subvolumes of the other bodies. In practice, the cost of this method does not vary substantially as the positions of the bodies are changed.

The algorithm has two primary shortcomings. First, by only checking for vertex-tetrahedra intersections, it is possible to miss some collision events, such as edge-edge collisions. Furthermore, it ignores intra-object collisions, which occasionally result from substantial model deformation.

## 3 Implementation

For real-time simulation, fast computation of the core procedures of the Finite Element method is crucial. We have achieved a ten-fold speedup by taking advantages of special properties of the strain and stress tensors.

### 3.1 Forces

Most of the computing time is spent within the computation of the nodal forces $\mathbf{f}_i$ based on their actual coordinates $\mathbf{x}_i$ in Eq. 9. A direct implementation requires $4 \times 4 \times 3 \times 3 \times 2 = 288$ multiplications. We reduce this number dramatically by first splitting the sum into two parts, the evaluation of $4 \times 4$ weights

$$\mathbf{W} = \beta \mathbf{G} \sigma \mathbf{G}^{\mathbf{T}} \beta^{\mathbf{T}} \tag{19}$$

with $\mathbf{G}$ as defined in Eq. 10, and the computation of the force components:

$$\mathbf{f}_i = \frac{\mathrm{vol}}{2} \mathbf{W} \mathbf{x}_i \tag{20}$$

Fist we note that the $\mathbf{f}_i$ are independent of the undeformed position and the orientation of the tetrahedron, which means that we can make $[\mathbf{m}_1, \ldots, \mathbf{m}_4]$ lower triangular via a rotation (as determined by a QR decomposition). This causes $\beta_{00}, \beta_{01}$ and $\beta_{10}$ to be zero. Then we precompute all products $\beta_{ij}\beta_{kl}$. By taking advantage of the symmetry in Eq. 19 and the zero entries, only 45 values need to be computed and stored. This computation can be performed before simulating, or whenever a new tetrahedron is generated.

Second, because the stress tensor is symmetric, we have $w_{ij} = w_{ji}$. The fact that $\beta_{00}, \beta_{01}$ and $\beta_{10}$ are zero cancels out most of the addends in Eq. 19. By unrolling all the loops, and making use of the above observations, we achieved a speedup of 10.

### 3.2 The Jacobian

For static analysis as well as for implicit integration in a dynamic analysis, the Jacobian $J$ of $F$ is needed. The $3N$ by $3N$ matrix $J$ can be computed by adding the local 12 by 12 matrices $J_e$ of each element. Also, if during the simulation tetrahedra are deleted or generated by the fracturing process, their matrices can be subtracted and added dynamically to the global matrix $J$. The components of $J_e$ are

$$f'_{ij} = \frac{\partial f_{ij}}{\partial x_{rs}}, \tag{21}$$

where $f_{ij}$ is the $j$th component of the force vector at the $i$th node of tetrahedron $e$ and $x_{rs}$ is the $s$th component of the deformed coordinate of node $r$ ($i, r \in 1 \ldots 4$ and $j, s \in 1 \ldots 3$).

The derivatives of the weights with respect to $x_{rs}$ are

$$\mathbf{W}' = \beta \mathbf{G} \sigma' \mathbf{G}^{\mathbf{T}} \beta^{\mathbf{T}}, \tag{22}$$

where $\sigma'$ is the the derivative of the stress tensor with respect to $x_{rs}$. For derivatives of the forces we get

$$f'_{ij} = \frac{\text{vol}}{2}(\mathbf{W}'\mathbf{x}_i + \mathbf{W}\mathbf{x}'_i) \tag{23}$$

Since both $\sigma'$ and $\mathbf{W}'$ are symmetric, the acceleration methods discussed in the previous section can also be applied to computation of the entries of $J$, and likewise results in a ten fold speedup.

## 4 Results

The following examples demonstrate that with our hybrid simulation technique, malleable and brittle objects can be animated in real-time without significant loss of realism. All animations are computed with rates in the range of 5 to 10 frames per second on an SGI Octane 2 (R12000, dual 400 MHz). The integration of the rigid body equations takes between 10 to 20 milliseconds per time step in all the examples. The time to compute deformation and fracture depends on the number of tetrahedra in the object and for our models (1000 - 4000 tetrahedra) varies between 10 and 80 milliseconds. The real-time system can also dynamically texture exposed surfaces without substantially impacting the frame rate. A video demonstration in AVI format can be downloaded from our webpage at graphics.lcs.mit.edu/simulation.

### 4.1 Vase

The frames from the animation sequences shown in Figure 3 (see Appendix) demonstrate brittle fracture of a china vase composed of 1440 tetrahedra striking the ground. Because of the material properties of the object, the vase fractures with only minimal deformation. Cracks grow instantaneously and separate the body into multiple new objects. The velocities and angular momenta of these objects are derived from the state of the original object as described in Section 2.7. Pictures (a) and (b) of Figure 6 show internal tensile stresses in shades of red.

### 4.2 Clay Teddy

Figure 4 shows a teddy bear modeled with 3747 tetrahedra. It is made of soft clay that deforms at the instant of impact. The deformations are computed as the static response to collision forces, which are absorbed by the material. After several hits (a), the bear's shape (c) deviates substantially from the undeformed model (b).

### 4.3 Cinder Blocks

Our third example demonstrates a real-time collision detection sequence in which one cinder block is dropped onto another (see Figure 5. Each block is modeled with 824 tetrahedra. Our system renders solid textures to the exposed surfaces of the blocks, dynamically generating new textures as additional faces are exposed by the fracture process. Figure 6(c) shows the internal stresses at the moment of impact.

## 5 Conclusions

We have described a fast method for simulating the deformation and fracture of malleable and brittle objects in real time. By employing a hybrid simulation strategy that alternates between a rigid body dynamics simulation and a continuum model at the point of impacts, we are able to compute robust solutions to otherwise stiff system equations. Our continuum model finds the static equilibrium of the system after all the initial transient behavior has settled out. The added information provided by this solution allows us to compute plausible deformations and fracturing of an interesting class of plastic and brittle materials.

One limitation of our system is that it only considers deformation and fracture behaviors at the instant of contact. We have also found that the problem of real-time collision detection for object in a near-contact state along a significant boundary, such along a fracture line, is at least as time-consuming as the system simulations. Furthermore, appropriate collision responses are extremely important in judging the realism of an animation.

We are excited by the performance of our current system and we are investigating a range of applications that might benefit from simulation approach. We are planning to use our system in a real-time sculpting environment, within which we are hoping to incorporate more dynamic simulation capabilities.

## References

1. D. Baraff and A. Witkin. Large steps in cloth simulation. In *Computer Graphics Proceedings*, Annual Conference Series, pages 43–54. ACM SIGGRAPH, August 1998.
2. K. J. Bathe. *Finite Element Procedures in Engineering Analysis*. Prentice-Hall, New Jersey, 1982.
3. T. J. Chung. *Applied Continuum Mechanics*. Cambridge Univ. Press, NY, 1996.
4. R. D. Cook. *Concepts and Applications of Finite Element Analysis*. John Wiley & Sons, NY, 1981.
5. E. E. Gdoutos. *Fracture Mechanics*. Kluwer Academic Publishers, Netherlands, 1993.
6. S. F. Gibson and B. Mitrich. *A survey of deformable models in computer graphics*. Technical Report TR-97-19, Mitsubishi Electric Research Laboratories, Cambridge, MA, 1997.
7. J. F. O'Brien and J. K. Hodgins. Graphical modeling and animation of brittle fracture. In *Computer Graphics Proceedings*, Annual Conference Series, pages 287–296. ACM SIGGRAPH, August 1999.
8. C. Pozrikidis. *Numerical Computation in Science and Engineering*. Oxford Univ. Press, NY, 1998.
9. J. Smith, A. Witkin, and D. Baraff. Fast and controllable simulation of the shattering of brittle objects. *Computer Graphics Interface*, pages 27–34, May 2000.
10. D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. In *Computer Graphics Proceedings*, Annual Conference Series, pages 269–278. ACM SIGGRAPH, August 1988.
11. D. Terzopoulos and A. Witkin. Physically based models with rigid and deformable components. *IEEE Computer Graphics & Applications*, pages 41–51, November 1988.
12. A. Witkin and D. Baraff. Physically based modeling: Principles and practice. *SIGGRAPH Course notes*, August 1997.
13. Y. Zhuang. *Real-time Simulation of Physically Realistic Global Deformation*. Ph. D. thesis of Univ. of California, CA, 2000.

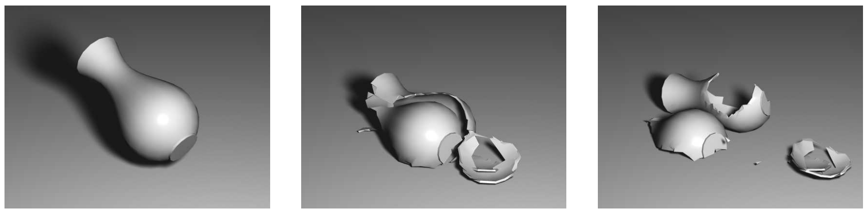

**Fig. 3.** Drop of a vase demonstrates brittle fracture

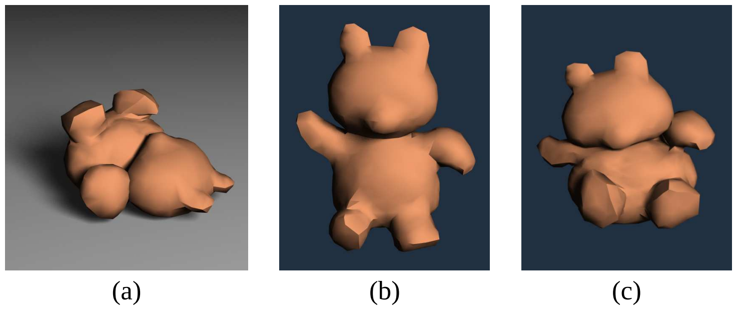

**Fig. 4.** Clay teddy bear after dropping (a), front view before (b) and after deformation (c)

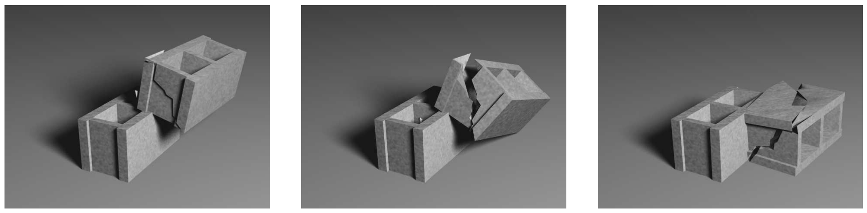

**Fig. 5.** Collision detection demonstrated with two cinder blocks

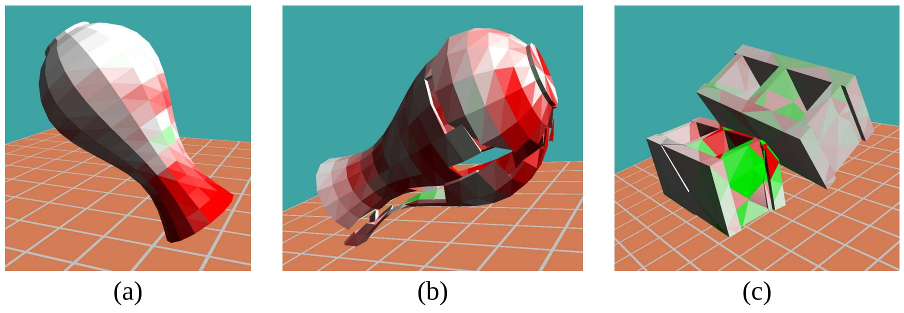

**Fig. 6.** Visualization of tensile stress within objects