Interactive Virtual Materials

Matthias Müller Markus Gross

ETH Zürich

Figure 1: The pitbull with its inflated head (left) shows the artifact of linear FEM under large rotational deformations. The correct deformation is shown on the right.

Abstract

In this paper we present a fast and robust approach for simulating elasto-plastic materials and fracture in real time. Our method extends the warped stiffness finite element approach for linear elasticity and combines it with a strain-state-based plasticity model. The internal principal stress components provided by the finite element computation are used to determine fracture locations and orientations. We also present a method to consistently animate and fracture a detailed surface mesh along with the underlying volumetric tetrahedral mesh. This multi-resolution strategy produces realistic animations of a wide spectrum of materials at interactive rates that have typically been simulated off-line thus far.

Key words: Physically Based Animation, Finite Element Method, Large Deformations, Stiffness Warping, Elasticity, Plasticity, Fracture.

1 Introduction

Today's interactive graphics applications, such as computer games or simulators, demand a continuously growing degree of visual realism and technical sophistication. This demand poses great challenges for the underlying real-time graphics algorithms including rendering and animation. In addition to the display quality, it is especially the way in which the physical behavior of characters, objects, or entire scenes are simulated that eventually determines the degree of realism experienced by the user. Using animation sequences predefined by an artist is a viable way to simulate physical effects in interactive systems. However, this method exhibits clear limitations when it comes to the realistic simulation of material behavior, such as elasticity, plasticity, melting, or fracture. Therefore, methods that control the object by physical laws have received increasing attention.

In the past decades, computational scientists have devised various methods for the simulation of material behavior with a high level of accuracy. The main goal of such simulations has been the authentic reproduction of the real world, whereas interactivity has not been a primary focus.

In contrast, in interactive graphics it is often not necessary to simulate the physical behavior with such numerical accuracy. Approximations are acceptable as long as they yield plausible and visually realistic results. Considerable research has been devoted to the development of such approximate simulation techniques. Going from off-line computations to interactive simulations, however, is not just a matter of making things faster. In order to allow for random interaction with the system, stability, visual quality and speed are the crucial requirements such methods have to satisfy.

In this paper, we present an unconditionally stable method to simulate a wide range of materials including elasticity, plasticity, melting, tearing and fracture in real time, while retaining geometric detail on the object's surface. Our method is primarily designed for applications in computer games, virtual surgery and related fields.

1.1 Related Work

During the last two decades, physically-based simulation methods tailored to the field of computer graphics have been developed. In the late eighties, Terzopoulos *et al.* proposed the use of physical models for animating elastic and plastic objects [21] and fracture effects [20] which pioneered the use of physically-based models in the field of computer graphics. They used surfaces to represent deformable objects and solved the governing partial differential equations using finite difference schemes. While their work focused on the correct modeling of physical effects, the speed of the simulation was of secondary importance and, at that time, computations were done off-line. In the following years, a variety of new models for linear and non-linear elastic objects were proposed in computer graphics (summarized in [8] and [22]) while

little attention was given to plasticity and fracture effects.

Recently, O'Brien *et al.* presented a finite-element-based technique for simulating brittle [16] and ductile [15] fracture in connection with elasto-plastic materials. Their method produces visually convincing results, but it is not designed for interactive or real-time use.

ArtDefo [10] designed by James and Pai was among the first systems to simulate elastic objects at interactive rates. They use a linear boundary element model with pre-computed modes for a fixed geometry. Debunne [4] and Wu [24] use a hierarchy of volumetric meshes and Grinspun [9] and James [11] a hierarchy of basis functions which speed up the simulation but make changes in a mesh caused by fracture difficult to handle. On the other hand, interactive fracture methods such as the ones proposed by Smith [19] and Müller [13] are tailored for rigid or brittle objects.

Since linear elasticity models are both stable and computationally cheap, they have become popular in the field of real-time animation. Linear models are, however, not suitable for large rotational deformations, because the non-linear effects create displeasing distortions of the geometry (see Fig. 1). To solve this problem Capell *et al.* [1] suggest manual division of an object into small parts based on its skeleton. Each part uses its local (rotated) coordinate frame to compute the linear elastic forces. The primary disadvantage of this approach is the discontinuity at the boundary between two parts.

An alternative to Capell's method is the warped stiffness approach proposed by Müller [12] which is a co-rotational formulation [6] and uses a local coordinate frame for each vertex to compute the linear force. The resulting discontinuities are smaller because individual rotations are computed per vertex. However, small local errors sum up and can cause ghost forces. For FEM-based cloth modeling, Etzmuss [5] improved the warped stiffness approach by using element-based rotation extraction.

1.2 Our Contribution

In this paper, we present a unified algorithmic framework for the real-time simulation of a variety of physical phenomena for geometrically complex objects. We propose a way to efficiently and robustly compute the elastic forces avoiding linear distortions and ghost forces. Both a plasticity and a fracture model are integrated into the pseudo-linear elastic force computation. We also propose an algorithm to consistently animate and fracture a high resolution surface mesh based on a coarser underlying volumetric mesh. This multi-resolution approach enables us to simulate geometrically complex objects in real time while simultaneously retaining their surface details.

2 Elasticity Model

Our elasticity model is based on the linear continuum elasticity theory [2]. We use the Finite Element Method (FEM) [3] with linear displacement tetrahedra to solve the governing partial differential equations. Here we summarize the steps essential to understand our method. We also include details of how the stiffness matrix is computed because some intermediate quantities are used in sections 3 and 4 to model plasticity and fracture.

2.1 The Stiffness Matrix

In continuum elasticity theory the deformation of an object is described by a vector field $\mathbf{u}(\mathbf{x}) = [u(\mathbf{x}), v(\mathbf{x}), w(\mathbf{x})]^T$ meaning that every point $\mathbf{x} = [x, y, z]^T$ in the undeformed body corresponds to point $\mathbf{x} + \mathbf{u}(\mathbf{x})$ in the deformed body. The first step in a finite element approach is to replace the continuous displacement field $\mathbf{u}(\mathbf{x})$ with a discrete set of displacement vectors defined only at the vertices of a mesh – in our case a tetrahedral mesh. Within each tetrahedral element e, the displacement field is linearly interpolated as

$$\mathbf{u}(\mathbf{x}) = \mathbf{H}_e(\mathbf{x}) \cdot \hat{\mathbf{u}}, \tag{1}$$

where $\mathbf{H}_e(\mathbf{x})$ is a 3×12 matrix that contains the shape functions of the tetrahedral element and $\hat{\mathbf{u}} = [u_1, v_1, w_1, \ldots, u_4, v_4, w_4]^T$ is the collection of the displacement vectors at the four vertices of the tetrahedron. Using Cauchy's linear strain tensor [3], for the strain within the tetrahedron we get

$$\epsilon = \mathbf{B}_e \cdot \hat{\mathbf{u}}, \tag{2}$$

where $\mathbf{B}_e$ a constant 6×12 matrix, which can be pre-computed for every tetrahedron. Using Hooke's law, for the stress within an element we get

$$\sigma = \mathbf{E} \cdot \epsilon = \mathbf{E}\mathbf{B}_e\hat{\mathbf{u}}, \tag{3}$$

where $\mathbf{E}$ is a 6×6 matrix which – for isotropic materials – only depends on two scalars, Young's modulus and Poisson's ratio [3]. The elastic forces $\mathbf{f}_e$ acting on the nodes of an element are derived from the strain energy which, in turn, depends on the strains and stresses within the element. Using Cauchy strain, the forces turn out to be linearly dependent on the vertex displacements $\hat{\mathbf{u}}$:

$$\mathbf{f}_e = \mathbf{K}_e\hat{\mathbf{u}} \tag{4}$$

where the 12×12 matrix $\mathbf{K}_e = V_e\mathbf{B}_e^T\mathbf{E}\mathbf{B}_e$ is the stiffness matrix and V_e the volume of the element. Finally, the $3n \times 3n$ dimensional stiffness matrix $\mathbf{K}$ of the entire mesh is an assembly of the individual $\mathbf{K}_e$ of all the elements.

2.2 Dynamic Deformation

To simulate the dynamic behavior of an object, the coordinate vector $\mathbf{x}$ is made a function of time, $\mathbf{x}(t)$. The following governing equation for $\mathbf{x}(t)$ in Lagrange's form describes the dynamics of the system:

$$\mathbf{M}\ddot{\mathbf{x}} + \mathbf{C}\dot{\mathbf{x}} + \mathbf{K}(\mathbf{x} - \mathbf{x}_0) = \mathbf{f}_{\text{ext}}, \tag{5}$$

where $\dot{\mathbf{x}}$ and $\ddot{\mathbf{x}}$ are the first and second derivatives of $\mathbf{x}$ with respect to time, $\mathbf{M}$ is the mass matrix, $\mathbf{C}$ the damping matrix and $\mathbf{f}_{\text{ext}}$ a vector of external forces. Eqn. (5) defines a coupled system of $3n$ linear ordinary differential equations for the n position vectors contained in $\mathbf{x}$.

The advantages of using linearized elastic forces are, first, that the stiffness matrix $\mathbf{K}$ is constant and can be pre-computed before the simulation starts. Second, when an implicit scheme is used to solve Eqn. (5), a purely linear system has to be solved at every time step. We choose implicit Euler integration because unlike explicit schemes it is unconditionally stable [23, 18]. Unconditional stability is essential in an interactive system. Third, if the tetrahedra in the original mesh are well shaped, the system matrix is well conditioned throughout the simulation. In contrast, when non-linear elastic forces are used, a stiffness matrix has to be computed at every time step and the system matrix can become arbitrarily ill-conditioned because it depends on the deformed model.

Linearized elastic forces are, however, only valid close to the equilibrium configuration. Under *large rotational deformations*, they cause unrealistic growth in volume as demonstrated by the pitbull model in Figure 1.

2.3 Element-Based Warped Stiffness

In [12] Müller *et al.*propose a method they call *Stiffness Warping* to remove the artifacts that linear elastic forces show while keeping the governing equation linear. They compute the elastic forces for every vertex in a local unrotated coordinate frame. However, the proposed way of extracting the rotational components of the deformation has two main disadvantages. First, the rotation $\mathbf{R}_i$ of vertex i has to be computed from the locations of its adjacent vertices which is an ambiguous problem. Second, the elastic forces are not guaranteed to sum up to zero resulting in possible ghost forces.

Following Etzmuss [5], we solve these problems by extracting rotations of elements rather than rotations of vertices. Contrary to the vertex-based approach, our method also allows the integration of the plasticity model described in the next section. For a single tetrahedral element with stiffness matrix $\mathbf{K}_e$, the forces $\mathbf{f}_e$ acting at its four vertices are

$$\mathbf{f}_e = \mathbf{K}_e \cdot (\mathbf{x} - \mathbf{x}_0) = \mathbf{K}_e \cdot \mathbf{x} + \mathbf{f}_{0e}, \tag{6}$$

where $\mathbf{x}$ contains the positions of the four vertices and $\mathbf{f}_{0e}$ contains force offsets. Now let us assume that we know the rotational part $\mathbf{R}_e$ of the deformation of the tetrahedron. Then, using the warped stiffness concept adopted to elements, we compute the forces as

$$\begin{aligned} \mathbf{f}_e &= \mathbf{R}_e\mathbf{K}_e \cdot (\mathbf{R}_e^{-1}\mathbf{x} - \mathbf{x}_0) \\ &= \mathbf{R}_e\mathbf{K}_e\mathbf{R}_e^{-1}\mathbf{x} - \mathbf{R}_e\mathbf{K}_e\mathbf{x}_0 \\ &= \mathbf{K}'_e\mathbf{x} + \mathbf{f}'_{0e}, \end{aligned} \tag{7}$$

where $\mathbf{R}_e$ is a 12×12 matrix that contains four copies of the 3×3 rotation matrix along its diagonal. By doing so, we reach the exact same forces as though we had computed the regular linear elastic forces in a rotated coordinate frame (see Fig. 2). The forces in $\mathbf{f}_e$ are, thus, guaranteed to sum to zero. Now, for the elastic forces of the entire mesh we get

$$\mathbf{f} = \mathbf{K}'\mathbf{x} + \mathbf{f}'_0, \tag{8}$$

where the global stiffness matrix $\mathbf{K}'$ and force offset vector $\mathbf{f}'_0$ are the sums of the element's *rotated* stiffness matrices $\mathbf{K}'_e = \mathbf{R}_e\mathbf{K}_e\mathbf{R}_e^{-1}$ and force offsets $\mathbf{f}'_{0e} = \mathbf{R}_e\mathbf{f}_{0e}$ with global vertex numbers. Our method has the following main features:

- *No ghost forces*: Since the individual $\mathbf{f}_e$ all sum to zero, the forces in $\mathbf{f}$ sum to zero, too, which means the method does not produce ghost forces.
- *Stability*: As in non-linear FEM, the global system matrix derived from Eqn. (5) changes for every time step. However, in non-linear FEM it can become arbitrarily ill-conditioned because it depends on the shape of the deformed tetrahedra. With our method, the local system matrices of elements are only rotated which does not change their condition. Since the global system matrix depends on the sum of differently rotated element matrices, its condition number might still change slightly. However, in all of our examples presented in section 7, the global condition number never changed by more than 25 percent. If the tetrahedra in the original mesh have good aspect ratios, the simulation remains stable for arbitrary deformations.
- *Speed:* In contrast to non-linear FEM, the element's stiffness matrices $\mathbf{K}_e$ can be pre-computed and reused saving a considerable amount of computation time [12]. However, unlike in the vertex-based technique, these matrices are rotated *before* the summation. This means that, while the $\mathbf{K}_e$ can still be pre-computed, the global stiffness matrix $\mathbf{K}'$ has to be summed up whenever the rotations change, i.e. at every time step. Fortunately, the time for the

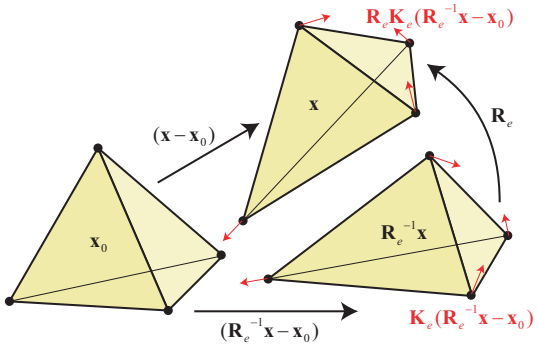

Figure 2: To compute the elastic forces acting at the vertices of a tetrahedron, its deformed coordinates $\mathbf{x}$ *are rotated back to an unrotated frame* $\mathbf{R}_e^{-1}\mathbf{x}$*. There the displacements* $\mathbf{R}_e^{-1}\mathbf{x} - \mathbf{x}_0$ *are multiplied with the stiffness matrix yielding the forces* $\mathbf{K}_e(\mathbf{R}_e^{-1}\mathbf{x} - \mathbf{x}_0)$ *that are finally rotated back to the frame of the deformed tetrahedron by multiplying them with* $\mathbf{R}_e$*.*

summation is linear in the number of elements and according to our experiments (Fig. 7), small in comparison to the time it takes to solve the linear system.

- *Quality*: The error of the approximation is the same as in linear FEM if the rotation matrices $\mathbf{R}_e = \mathbf{I}$ for the entire simulation. If all the $\mathbf{R}_e$ are the same but follow the rigid body transformation of the entire mesh, our method outperforms linear FEM because it produces the same forces as if the linear model was re-computed in the rotated frame for every time step and, thus, correctly produces zero forces for pure rigid body transformations. When individual $\mathbf{R}_e$'s are computed for every element, the approximation gets even better because it can then adopt to local rotations.

2.4 Rotation of a Tetrahedron

There is a simple way to compute rotations $\mathbf{R}_e$ of tetrahedral elements. Let us look at a tetrahedron whose vertices have coordinates $\mathbf{p}_1, \ldots, \mathbf{p}_4$ in the undeformed state and $\mathbf{q}_1, \ldots, \mathbf{q}_4$ in the deformed state. The barycentric coordinates $\beta_1, \ldots, \beta_4$ of a point $\mathbf{p}$ with respect to the undeformed tetrahedron satisfy

$$\begin{bmatrix} p_{1x} & p_{2x} & p_{3x} & p_{4x} \\ p_{1y} & p_{2y} & p_{3y} & p_{4y} \\ p_{1z} & p_{2z} & p_{3z} & p_{4z} \\ 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} \tag{9}$$

or $\mathbf{P}\beta = \mathbf{p}$. The point $\mathbf{q}$ in the deformed tetrahedron that corresponds to $\mathbf{p}$ has the same barycentric coordinates β but with respect to the deformed tetrahedron

$$\mathbf{Q}\beta = \mathbf{q} \tag{10}$$

and, thus, combining Eqn. 9 and 10 we get

$$\mathbf{q} = \mathbf{Q}\beta = \mathbf{Q}\mathbf{P}^{-1}\mathbf{p} = \mathbf{A}\mathbf{p}. \tag{11}$$

The unique matrix $\mathbf{A} = \mathbf{Q}\mathbf{P}^{-1}$ that describes the transformation of the tetrahedron has the form

$$\mathbf{A} = \begin{bmatrix} & \mathbf{B} & & \mathbf{t} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{12}$$

where $\mathbf{t}$ contains the translational part of the transformation and $\mathbf{B}$ the rotation and stretching parts. The 3 by 3 matrix $\mathbf{B}$ is independent of translations of both $\mathbf{P}$ and $\mathbf{Q}$. Finally, the rotational part of the transformation can be extracted by a polar decomposition of $\mathbf{B}$ as proposed in [5].

3 Plasticity Model

So far, our model is perfectly elastic, i.e. when external forces are removed, the model returns to its original shape. In contrast, an elasto-plastic material stores part of the deformation and returns to a configuration between the deformed and undeformed state when the external forces are removed. In [15] O'Brien describes a method for modelling plasticity. The method is used in the context of non-linear FEM and explicit integration. Here we show how it can be adopted to our framework, namely linear, warped FEM in connection with implicit integration. We also suggest a way to simulate plastic creep, an effect not present in O'Brien's model.

According to Eqn. (2) and our warped stiffness approach, a deformed tetrahedon is under a total strain of

$$\epsilon_{\text{total}} = \mathbf{B}_e \cdot \hat{\mathbf{u}} = \mathbf{B}_e(\mathbf{R}_e^{-1}\mathbf{x} - \mathbf{x}_0). \tag{13}$$

A plastic element stores plastic strain in a state variable $\epsilon_{\text{plastic}}$ and only the difference between the total strain and the plastic strain – the elastic strain – causes internal elastic forces (see Fig. 3):

$$\epsilon_{\text{elastic}} = \epsilon_{\text{total}} - \epsilon_{\text{plastic}} \tag{14}$$

Note that these strains are six-dimensional vectors. The state variable $\epsilon_{\text{plastic}}$ is initialized with $\mathbf{0}$. At every time step, it is updated as follows:

$\epsilon_{\text{total}} \leftarrow \mathbf{B}_e(\mathbf{R}_e^{-1}\mathbf{x} - \mathbf{x}_0)$
$\epsilon_{\text{elastic}} \leftarrow \epsilon_{\text{total}} - \epsilon_{\text{plastic}}$
if $||\epsilon_{\text{elastic}}||_2 > c_{\text{yield}}$: $\epsilon_{\text{plastic}}$ += $\Delta t \cdot c_{\text{creep}} \cdot \epsilon_{\text{elastic}}$
if $||\epsilon_{\text{plastic}}||_2 > c_{\text{max}}$: $\epsilon_{\text{plastic}}$ *= $c_{\text{max}}/||\epsilon_{\text{plastic}}||_2$

First, the total and elastic strains are updated. The plasticity model has three scalar parameters $c_{\text{yield}}, c_{\text{creep}}$ and

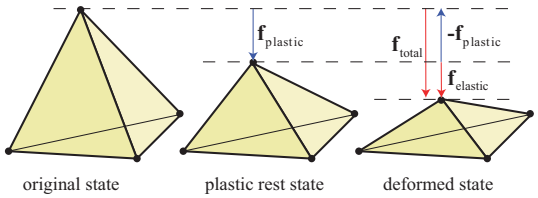

Figure 3: Relative to the original state the deformation of the tetrahedron yields a force of $\mathbf{f}_{total}$. In a plastic material only the offset force $\mathbf{f}_{total} - \mathbf{f}_{plastic}$ relative to the plastic rest state acts as elastic force $\mathbf{f}_{elastic}$. Here, the stiffness is chosen such that displacement equals force.

c_{max}. If the 2-norm of the elastic strain exceeds the threshold c_{yield}, the plastic strain absorbs part of it. If $c_{\text{creep}} \in [0 \dots 1/\Delta t]$ is $1/\Delta t$, the elastic strain is immediately and completely absorbed. Small values for c_{creep} yield slow plastic flow in the material. The parameter c_{max} defines the maximum plastic strain an element can store. If the 2-norm of the plastic strain exceeds c_{max}, the plastic strain is scaled down accordingly.

We integrate the effect of the plastic strain into Eqn. (5) via corresponding plastic forces. This way the stiffness matrices $\mathbf{K}_e$ of the elements do not have to be recomputed. We use the definition of the stiffness matrix from Eqn. (4) to compute the plastic forces $\mathbf{f}_{\text{plastic}}$ that correspond to $\epsilon_{\text{plastic}}$:

$$\begin{aligned} \mathbf{f}_{\text{plastic}} &= \mathbf{R}_e \mathbf{K}_e \hat{\mathbf{u}}_{\text{plastic}} \\ &= \mathbf{R}_e \mathbf{K}_e \mathbf{B}_e^{-1} \cdot \epsilon_{\text{plastic}} \\ &= \mathbf{R}_e (V_e \mathbf{B}_e^T \mathbf{E} \mathbf{B}_e) \mathbf{B}_e^{-1} \cdot \epsilon_{\text{plastic}} \\ &= \mathbf{R}_e V_e \mathbf{B}_e^T \mathbf{E} \cdot \epsilon_{\text{plastic}} \\ &= \mathbf{R}_e \mathbf{P}_e \cdot \epsilon_{\text{plastic}} \end{aligned} \tag{15}$$

The plasticity matrix $\mathbf{P}_e = V_e \mathbf{B}_e^T \mathbf{E}$ that maps the plastic strain to plastic forces is constant and can be precomputed for each element. At every time step we compute the sum of the plastic forces coming from all elements and simply subtract them from Eqn. 5 to get

$$\mathbf{M}\ddot{\mathbf{x}} + \mathbf{C}\dot{\mathbf{x}} + \mathbf{K}'\mathbf{x} + \mathbf{f}_0' - \mathbf{f}_{\text{plastic}} = \mathbf{f}_{\text{ext}}. \tag{16}$$

Since we only add a force vector $\mathbf{f}_{\text{plastic}}$ to the equation of motion at every time step, plasticity does not change the condition of the linear system which needs to be solved.

4 Fracture Model

To complete our model, we propose a method, similar to the ones described in [13] and [16], for animating material tearing and fracture if internal stresses exceed a fracture threshold.

The concept behind our fracture algorithm is to find the maximum tensile stress σ_{max} and its direction $\mathbf{n}_{max}$ for every tetrahedral element and to fracture the model normal to $\mathbf{n}_{max}$ if σ_{max} exceeds the threshold of the material. Thus, at every time step we compute, for each tetrahedron, the stress tensor $\sigma = \mathbf{E}\mathbf{B}_e(\mathbf{R}_e^{-1}\mathbf{x} - \mathbf{x}_0)$, its largest eigenvalue σ_{max} and its corresponding eigenvector $\mathbf{n}_{max}$. If σ_{max} exceeds the fracture threshold of the material, we split the mesh as follows (see Fig. 4): First, we select randomly one of the tetrahedron's vertices that is marked as a crack tip. If none of them are marked, we randomly make a choice amongst all four vertices. We do this for two reasons. First, because it is more likely that an existing crack propagates than that a new one is formed and second, because in real materials, randomly located microscopic imperfections are the points where cracks are initiated [7].

Once a vertex v is selected to be split, we virtually put a plane α through v perpendicular to $\mathbf{n}_{max}$. Then v is split into two new vertices v_+ and v_-. For all tetrahedra adjacent to v, their vertex v is replaced by either v_+ or v_- depending on whether their center lies on the positive or negative side of α. For all pairs of tetrahedra lying on opposite sides of α who shared a common face (v, v_1, v_2) before the split, two sides (v_+, v_1, v_2) and (v_-, v_1, v_2) get exposed after the split and v_1 and v_2 are marked as crack tips.

With this method the cracks can only go along the tetrahedral boundaries which can result in visual artifacts. A way to reduce them is to use a non-regular tetrahedral mesh. We use this simple, fracture procedure for two main reasons. First, it is very fast because the manipulations in the mesh are minimal. Second, the stiffness matrices $\mathbf{K}_e$ of the elements do not change. The way they are added to the global stiffness matrix does change but the summation is done at every time step anyway. Fracture, thus, causes no computational overhead other than some updates in the adjacency lists of the mesh. Also, as with the plasticity modifications, fracture does not introduce any instability to the linear system to be solved.

5 Surface Mesh Animation

Even though our method is fast enough to animate a few thousand volumetric elements at interactive rates (Fig. 7), a tetrahedral mesh with a resolution of this order cannot represent a surface of appealing quality. A natural way to solve this problem is to work with two different representations for the same object, a low resolution volumetric mesh for the FEM simulation and a high resolution surface mesh for rendering. This is a reasonable solution because firstly, surface detail does not significantly affect the physical behavior of an object and secondly, the deformation field stored in the low resolution volumetric mesh provides enough information to animate a detailed

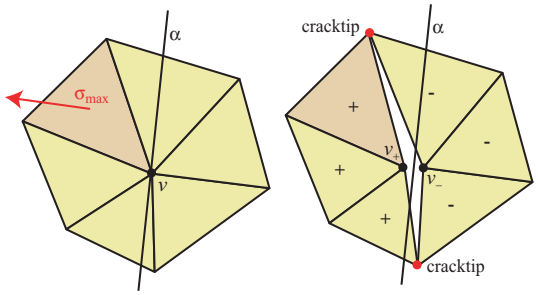

Figure 4: Planar sketch of a vertex split in 3-d. To fracture a tetrahedron, an adjacent vertex v is chosen. Plane α is placed perpendicular to the maximum stress component σ_{max} through v. Then v is split into v_+ and v_-. Tetrahedra adjacent to v that lie on the positive side of α are linked to v_+, the others to v_-.

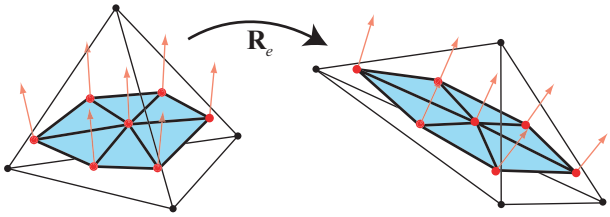

Figure 5: Vertices of the surface mesh (shown in red) are displaced according to the displacement field of the tetrahedron in which they lie. The rotation of the tetrahedron $\mathbf{R}_e$ is used to rotate the normals.

surface mesh. Pentland et al. [17] made use of this general idea by approximating deformation modes with polynomial mappings. The approach we propose here is more closely related to the work of Capell et al. [1] where a coarse control mesh is used to compute dynamic deformations of detailed surfaces.

5.1 Mesh Coupling

To animate a surface mesh consistently with a volumetric mesh, we first link every vertex of the surface mesh to the closest tetrahedron in the volumetric mesh and store its barycentric coordinates with respect to that tetrahedron. During the simulation, the position of each vertex of the surface mesh is interpolated from the positions of the linked tetrahedron using the stored barycentric coordinates. An advantage of the warped stiffness algorithm is that it computes the rotation $\mathbf{R}_e$ of every tetrahedral element which can directly be used to transform the normals stored in the undeformed surface mesh (see Fig. 5). This simple algorithm works very well with elastic and elasto-plastic deformation (see Figures 1 and 8).

5.2 Consistent Watertight Fracturing

When the volumetric mesh fractures as described in section 4, the mesh coupling becomes more complicated. There are two main problems that have to be solved:

- The vertices of the surface mesh and those of the volumetric mesh are not aligned. The three vertices of a surface triangle can be linked to different tetrahedra that can get detached by the fracture procedure. Such a triangle has to be split to prevent the surface from getting stretched like rubber across the crack of the volumetric mesh.

- If a two-dimensional surface mesh is fractured, unacceptable holes will appear.

The method that we propose solves the aforementioned problems and keeps an initially watertight surface mesh watertight throughout the entire simulation. The two requirement for the procedure to work correctly are, first, that the volumetric mesh *totally* contains the surface mesh and second, that the surface mesh is a manifold.

The basic event in the fracture procedure of the volumetric mesh that triggers surface updates is the disconnection of two tetrahedra that share a face (v, v_1, v_2) when vertex v is split. In this case two new faces (v_+, v_1, v_2) and $(v-, v_1, v_2)$ with identical coordinates are created in the volumetric mesh. The surface mesh is updated in two steps (see Fig. 6).

First, it is cut along triangle $T_v = (v, v_1, v_2)$, the common face along which the two tetrahedra are separated. If T_v does not intersect the surface, this step is skipped.

The cut operation can be done using the undeformed coordinates of both the volumetric and the surface mesh. We use a uniform spatial grid to quickly find the triangles of the surface mesh that intersect triangle T_v. By working with undeformed coordinates, this grid does not need to be updated when the object deforms.

In a second step, the holes in the surface are closed. Therefore a new surface triangle at the location of T_v is generated. This triangle is linked to the first tetrahedron and a copy of it to the second one. These two triangles close the hole. In the event that the new triangles reside completely inside the surface, the second step is finished. However, if T_v intersects the surface, the new triangles are subdivided along the surface mesh resulting in a set of smaller triangles of which those who lie outside the surface are deleted.

6 The Algorithm

Now we are ready to put everything together and summarize the entire simulation algorithm:

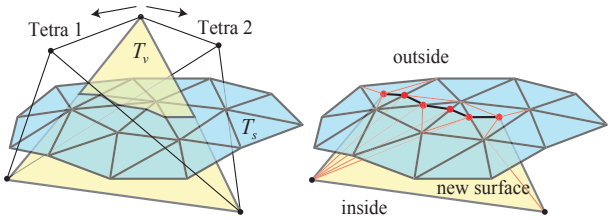

Figure 6: The surface mesh needs to be updated when two tetrahedra that share side T_v are detached. First, the surface mesh is cut along T_v. Then, a new surface triangle congruent with T_v is subdivided along the surface mesh and the triangles that are outside the surface are removed. The ones that are inside are linked to the first tetrahedron and identical copies to the second one.

forall elements e compute $\mathbf{B}_e, \mathbf{P}_e, \mathbf{K}_e$ from $\mathbf{x}_0$
initialize $\mathbf{x}^0, \mathbf{v}^0$
$i \leftarrow 0$
loop
  forall elements e compute $\mathbf{R}_e$ based on $\mathbf{x}^i$
  forall elements e update $\epsilon_{\text{plastic},e}$
  assemble $\mathbf{K} = \sum_e \mathbf{R}_e \mathbf{K}_e \mathbf{R}_e^{-1}$
  assemble $\mathbf{f}_0 = -\sum_e \mathbf{R}_e \mathbf{K}_e \mathbf{x}_0$
  assemble $\mathbf{f}_{\text{plastic}} = \sum_e \mathbf{R}_e \mathbf{P}_e \epsilon_{\text{plastic},e}$
  compute external forces $\mathbf{f}_{\text{ext}}$
  solve $(\mathbf{M} + \Delta t \mathbf{C} + \Delta t^2 \mathbf{K})\mathbf{v}^{i+1} =$
    $\mathbf{M}\mathbf{v}_i - \Delta t(\mathbf{K}\mathbf{x}^i + \mathbf{f}_0 + \mathbf{f}_{\text{plastic}} - \mathbf{f}_{\text{ext}})$
    for $\mathbf{v}^{i+1}$
  update $\mathbf{x}^{i+1} = \mathbf{x}^i + \Delta t \mathbf{v}^{i+1}$
  forall elements e
    compute stress $\sigma = \mathbf{E}\mathbf{B}_e(\mathbf{R}_e^{-1}\mathbf{x} - \mathbf{x}_0)$
    if maximum eigenvalue of $\sigma >$ fracture threshold
      then fracture mesh
  endfor
  $i \leftarrow i + 1$
endloop

7 Results

All the animations described in this section were computed and rendered in real time on a 1.8 GHz Pentium IV PC with a GForce 4 graphics card. For implicit integration we used a fixed time step of 0.01 seconds. To generate tetrahedral meshes from surface meshes we used the approach described in [14].

The computational cost of one time step dependent on the number of animated tetrahedra for the linear and warped stiffness approach is depicted in Figure 7. The curves show that the additional work of computing rotations and assembling the stiffness matrix is small in comparison to the task of solving the linear system. We use a conjugate gradients solver with a fixed number of iterations (20 for our demos).

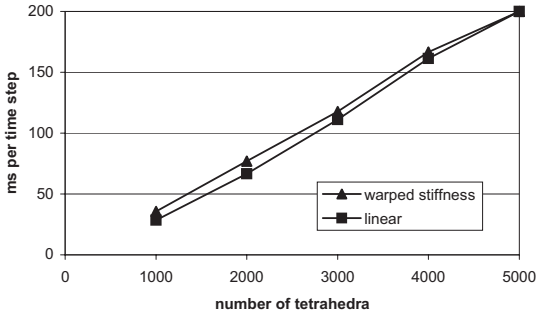

Figure 7: Computation time [ms] per time step against size of the mesh for warped stiffness and linear FEM.

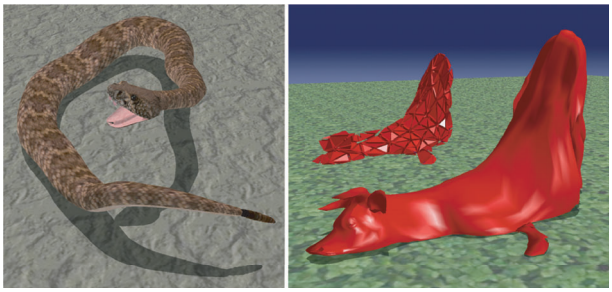

Figure 8: The snake on the left demonstrates large rotational elastic deformations while the cow made of warm wax melts plastically under gravity.

Figures 8 and 9 show the range of phenomena that can be simulated at interactive rates with the method described in this paper from large, purely elastic deformations (snake model), plastic deformation of warm wax under gravity (cow), local plastic deformation caused by a heat source (dragon) to fracture effects caused by user applied forces. The parameters used for these animations are summarized in Table 1. We made the entire simulation framework with a variety of additional scenes and a video available from www.matthiasmueller.info/demos.

8 Conclusions

We have presented a novel method for the real-time simulation of a wide range of material effects including elasticity, plasticity, melting and fracture. The key to our technique is a pseudo-linear elasticity model utilizing the warped stiffness approach, which was extended to avoid ghost forces and to make it amenable to plasticity and fracture modeling. To enhance the visual quality of our simulations, we developed algorithms to consis-

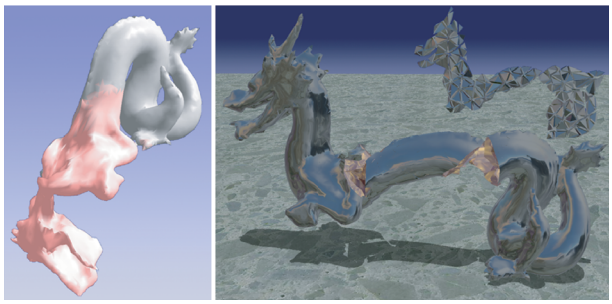

Figure 9: A heat source changes the material properties locally (left). The surface mesh of the dragon fractures along with the volumetric mesh when the model is pulled apart (right).

Model parameters	Snake	Cow	Dragon
Tetrahedra	440	970	834
Triangles	22,000	5,800	10,000
Elast. Mod. $[1e3N/m^2]$	100	50	100
Poisson Ratio $[1]$	0.33	0.33	0.33
Density $[kg/m^3]$	1000	1000	1000
Yield stress $[1e3N/m^2]$	∞	∞	60
Plastic yield $[1]$	∞	0.01	0.10
Plastic creep $[1/s]$	0	100	1
Plastic max $[1]$	0	10	1

Table 1: Parameters used for the animations.

tently combine a high resolution skin mesh with a computational volumetric mesh of lower resolution keeping the overall representation watertight at all times. Our method is easy to implement, stable and, as we demonstrate, performs robustly even in cases of extreme parameter settings. In terms of material stiffness, our method covers the gap between purely rigid objects – for which rigid body simulators are best suited – and fluids that change their topology because no connectivity is defined. In the future, we plan to extend our algorithms towards fluid flow simulation by including object representations that can handle changes in topology.

9 Acknowledgements

This project was funded by the Swiss National Commission for Technology and Innovation (KTI) project no. 6310.1 KTS-ET.

References

[1] S. Capell, S. Green, B. Curless, T. Duchamp, and Z. Popovic. Interactive skeleton-driven dynamic deformations. *Proceedings of ACM SIGGRAPH*, pages 586–593, 2002.

[2] T. J. Chung. *Applied Continuum Mechanics*. Cambridge Univ. Press, NY, 1996.

[3] R. D. Cook. *Finite Element Modeling for Stress Analysis*. John Wiley & Sons, NY, 1995.

[4] G. Debunne, M. Desbrun, M. P. Cani, and A. H. Barr. Dynamic real-time deformations using space & time adaptive sampling. *Proceedings of ACM SIGGRAPH*, pages 31–36, 2001.

[5] O. Etzmuss, M. Keckeisen, and W. Straßer. A fast finite element solution for cloth modelling. *Proceedings of Pacific Graphics 2003*, 2003.

[6] C. A. Felippa. *A Systematic Approach to the Element-Independent Corotational Dynamics of Finite Elements*. Report CU-CAS-00-03, Center for Aerospace Structures, Colorado, 2000.

[7] E. E. Gdoutos. *Fracture Mechanics*. Kluwer Academic Publishers, Netherlands, 1993.

[8] 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.

[9] E. Grinspun, P. Krysl, and P. Schroder. Charms: A simple framework for adaptive simulation. *Proceedings of ACM SIGGRAPH*, pages 281–290, 2002.

[10] D. James and D. K. Pai. Artdefo, accurate real time deformable objects. *Proceedings of ACM SIGGRAPH*, pages 65–72, 1999.

[11] D. James and D. K. Pai. Multiresolution green's function methods for interactive simulation of large-scale elastostatic objects. *ACM Transactions on Graphics*, 22(1):47–82, 2003.

[12] M. Müller, J. Dorsey, L. McMillan, R. Jagnow, and B. Cutler. Stable real-time deformations. *Proceedings of ACM SIGGRAPH Symposium on Computer Animation*, pages 49–54, 2002.

[13] M. Müller, L. McMillan, J. Dorsey, and R. Jagnow. Real-time simulation of deformation and fracture of stiff materials. *EUROGRAPHICS 2001 Computer Animation and Simulation Workshop*, pages 27–34, 2001.

[14] M. Müller and M. Teschner. Volumetric meshes for real-time medical simulations. *in Proceedings of BVM*, pages 279–283, 2003.

[15] J. F. O'Brien, A. W. Bargteil, and J. K. Hodgins. Graphical modeling and animation of ductile fracture. *Proceedings of ACM SIGGRAPH*, pages 291–294, 2002.

[16] J. F. O'Brien and J. K. Hodgins. Graphical modeling and animation of brittle fracture. *Proceedings of ACM SIGGRAPH*, pages 287–296, 1999.

[17] A. Pentland and J. Williams. Good vibrations: Modal dynamics for graphics and animation. *ACM Computer Graphics*, 23(3):215–222, 1989.

[18] C. Pozrikidis. *Numerical Computation in Science and Engineering*. Oxford Univ. Press, NY, 1998.

[19] 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.

[20] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: viscolelasticity, plasticity, fracture. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 269–278. ACM Press, 1988.

[21] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *Proceedings of ACM SIGGRAPH*, pages 205–214, 1987.

[22] A. Witkin and D. Baraff. Physically based modeling: Principles and practice. *Siggraph Course Notes*, August 1997.

[23] A. Witkin and D. Baraff. Large steps in cloth simulation. *Proceedings of ACM SIGGRAPH*, pages 43–54, 1998.

[24] X. Wu, M. S. Downes, T. Goktekin, and F. Tendick. Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. *Eurographics*, pages 349–358, September 2001.