Meshless Animation of Fracturing Solids

Mark Pauly, Stanford University
Richard Keiser, ETH Zurich
Bart Adams, KU Leuven
Philip Dutré, KU Leuven
Markus Gross, ETH Zurich
Leonidas J. Guibas, Stanford University

Figure 1: Brittle fracture of a hollow stone sculpture. Forces acting on the interior create stresses that cause the model to fracture and explode.

Abstract

We present a new meshless animation framework for elastic and plastic materials that fracture. Central to our method is a highly dynamic surface and volume sampling method that supports arbitrary crack initiation, propagation, and termination, while avoiding many of the stability problems of traditional mesh-based techniques. We explicitly model advancing crack fronts and associated fracture surfaces embedded in the simulation volume. When cutting through the material, crack fronts directly affect the coupling between simulation nodes, requiring a dynamic adaptation of the nodal shape functions. We show how local visibility tests and dynamic caching lead to an efficient implementation of these effects based on point collocation. Complex fracture patterns of interacting and branching cracks are handled using a small set of topological operations for splitting, merging, and terminating crack fronts. This allows continuous propagation of cracks with highly detailed fracture surfaces, independent of the spatial resolution of the simulation nodes, and provides effective mechanisms for controlling fracture paths. We demonstrate our method for a wide range of materials, from stiff elastic to highly plastic objects that exhibit brittle and/or ductile fracture.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

Keywords: physics-based animation, elasticity, plasticity, fracture, meshless methods

1 Introduction

Physics-based simulation has gained increasing importance in many fields of computer graphics, including 3D game engines, computer animation for feature films, surgery simulation, and virtual reality. With growing demand for realism and detail, the complexity of animations has been steadily increasing, incorporating a wide range of physical phenomena, such as elastic and plastic deformation [Terzopoulos et al. 1987], [Terzopoulos and Fleischer 1988], melting and flow [Carlson et al. 2002], fire [Nguyen et al. 2002], smoke [Fedkiw et al. 2001], and explosions [Feldman et al. 2003]. A central goal of physics-based animation is to ease the burden of the animator by partially automating the creation of particularly complex animations that are consistent with our everyday experiences of the physical world. Fracture simulation of deforming solids [O'Brien and Hodgins 1999], [O'Brien et al. 2002] is one prominent example, where manually specifying all animation parameters quickly becomes infeasible. Research in computer animation has thus focused on simulating fracturing materials based on physical models developed in computational fracture mechanics [Anderson 1995].

A current trend in this area is to enhance finite element methods (FEMs) using *meshfree methods*, e.g., particle-based approaches, to model the physical behavior around a crack front [Sukumar et al. 2000], [Belytschko et al. 2003] or to resort entirely to meshless methods [Krysl and Belytschko 1999]. There are a number of features of these methods that make them favorable for fracture simulation (see [Belytschko et al. 1996] for a detailed discussion). Most importantly, meshless methods avoid complex remeshing operations and the associated problems of element cutting and mesh alignment sensitivity common in FEM. Maintaining a conforming mesh can be a notoriously difficult task when the topology of the simulation domain changes frequently [Ortiz and Pandolfi 1999]. Repeated remeshing operations can adversely affect the stability and accuracy of the calculations, imposing undesirable restrictions on the time step. Finally, meshless methods are well suited for handling large deformations due to their flexibility when locally refining the sampling resolution.

On the downside, special treatment is required for the enforcement of essential boundary conditions, due to the lack of the Kronecker delta property of meshless shape functions (see [Fernandez-Mendez and Huerta 2004] for a recent survey). Also, meshless methods are computationally more involved, since the connectivity of nodes is determined at run-time and the evaluation of the shape functions requires an inversion of the moment matrix. However, by exploiting temporal coherence using local caching schemes, the computational burden can be reduced significantly as has lately been demonstrated in the real-time system of [Müller et al. 2004].

1.1 Contributions

In this paper we are concerned with the application of meshless methods to the domain of computer animation. The meshless approach provides us with essential flexibility in adapting the volume and surface sampling resolutions to the simulation fidelity and appearance requirements of the animation, while handling topological changes in a lightweight and efficient manner. Our key contribution

is a new meshless animation framework for elastic and plastic materials that fracture. Our method incorporates

- the dynamic creation and maintenance of fracture surfaces by continuously adding surface samples during crack propagation,
- a meshless initial sampling of the volumetric domain and local dynamic re-sampling that adapts the nodal sampling resolution to handle fracturing and large deformations,
- the dynamic adaptation of shape functions wherever new crack surfaces are created,
- the handling of the complex topological events associated with multiple branching and merging cracks.

After discussing related work in Section 1.2, we explain our meshless approach for solving the equations of continuum mechanics in Section 2. Section 3 describes how we propagate cracks through a solid and control the topology of crack surfaces and the simulation domain. Dynamic adaptation of the spatial discretization is discussed in Section 4 and details of the implementation are given in Section 5. We conclude the paper with a discussion of results and an outlook on future work.

1.2 Relation with Previous Work

[Terzopoulos et al. 1987] pioneered physics-based animation of deforming objects using finite difference schemes to solve the underlying elasticity equations. This work has been extended in [Terzopoulos and Fleischer 1988] to handle plastic materials and fracture effects. Mass-spring models [Hirota et al. 1998] and constraint-based methods [Smith et al. 2000] have also been popular for modeling fracture in graphics, as they allow for easy control of fracture patterns and relatively simple and fast implementations. Recent efforts have focused on finite element methods that directly approximate the equations of continuum mechanics [Chung 1996]. O'Brien et al. were the first to apply this technique for graphical animation in their seminal paper on brittle fracture [O'Brien and Hodgins 1999]. Using element cutting and dynamic remeshing they adapt the simulation domain to conform with the fracture lines that are derived from the principal stresses. [O'Brien et al. 2002] introduces strain state variables to model plastic deformations and ductile fracture effects. Element splitting has also been used in virtual surgery simulation, where [Bielser et al. 2003] introduced a state machine to model all configurations of how a tetrahedron can be split. [Müller et al. 2001] and [Müller and Gross 2004] demonstrate real-time fracturing using an embedded boundary surface to reduce the complexity of the finite element mesh. The virtual node algorithm of Molino et al. [2004] combines the ideas of embedding the surface and remeshing the domain. Elements are duplicated and fracture surfaces are embedded in the copied tetrahedra. This allows more flexible fracture paths, but avoids the complexity of full remeshing and associated time stepping restrictions.

Meshless methods have been introduced to computer graphics by Desbrun and Cani [1995], who model soft inelastic materials using particle systems coated with a smooth iso-surface. Smoothed Particle Hydrodynamics (SPH) has been applied in [Desbrun and Cani 1996] and extended using space and time adaptive sampling in [Desbrun and Cani 1999] (see also [Debunne et al. 2001]). [Chang and Zhang 2004] presents a meshless method for animating elastic solids, using linear superposition of approximate analytical solutions for point loading, analogous to the boundary element method of [James and Pai 1999].

Our system is mainly motivated by recent advances in meshless methods for computational mechanics (see [Liu 2002] or [Fries and Matthies 2003] for good overviews). We build upon [Müller et al. 2004] who introduced a meshless animation framework using point-based representations for both the simulation volume and the boundary surface of elastically and plastically deforming solids. To simulate fracture effects we integrate the dynamic computation of shape functions of [Belytschko et al. 1994] and [Organ et al. 1996], who discussed numerical examples for simple 2D crack problems. An extension to 3D domains was presented in [Krysl and Belytschko 1999], where fracturing was limited to non-interacting, non-branching cracks. [Ventura et al. 2002] introduced a meshless fracturing method using level sets to define fracture surfaces implicitly. They show examples for single cracks in 2D, but assert that their method can be extended to 3D simulations. In our scheme 3D fracture surfaces are modeled and sampled explicitly. This supports efficient modeling of fine geometric detail, such as sharp creases and corners, and allows for easy control of fracture patterns, both of which are crucial for graphics applications.

2 Meshless Simulation

We first give a brief review of the relevant equations of continuum mechanics and discuss the discretization in the meshless setting. Then we explain how we can adapt the simulation to incorporate discontinuities introduced by fracture.

Given a 3-dimensional solid with material coordinates $\mathbf{x}$, we define the positions of the deformed model in world coordinates as $\mathbf{x}+\mathbf{u}$, where $\mathbf{u} = (u,v,w)^T$ is a displacement vector field. The deformation induced by $\mathbf{u}$ creates a strain, which can be computed from the gradient $\nabla\mathbf{u}$ using the quadratic Green-Saint-Venant strain tensor as

$$\varepsilon = (\nabla\mathbf{u} + \nabla\mathbf{u}^T + \nabla\mathbf{u}\nabla\mathbf{u}^T)/2.$$

Assuming a Hookean material, the elastic stress σ is related to the strain ε as $\sigma = \mathbf{C}\varepsilon$, where $\mathbf{C}$ is a rank four tensor that defines the constitutive law of the material. The elastic force per unit volume at $\mathbf{x}$ can then be derived as $\mathbf{f} = -\sigma\nabla_{\mathbf{u}}\varepsilon$ (see [Müller et al. 2004] for details). To discretize the force distribution, the displacement field $\mathbf{u}$ is typically approximated as $\mathbf{u}(\mathbf{x}) \approx \sum_i \Phi_i(\mathbf{x})\mathbf{u}_i$, where $\mathbf{u}_i$ are the displacement vectors at a discrete set of nodes $\{\mathbf{x}_i\}$ and Φ_i are shape functions associated with these nodes. For FEM, the Φ_i are constructed using a tessellation of the simulation domain into non-overlapping elements. Meshless methods require no such spatial decomposition, but instead use techniques such as the moving least squares (MLS) approximation [Lancaster and Salkauskas 1981] to define the shape functions based on the location of the nodes only. Given a complete polynomial basis $\mathbf{p}(\mathbf{x}) = [1 \;\; \mathbf{x} \;\; \ldots \;\; \mathbf{x}^n]^T$ of order n and a weight function ω_i, the meshless shape functions can be derived as

$$\Phi_i(\mathbf{x}) = \omega_i(\mathbf{x},\mathbf{x}_i)\mathbf{p}^T(\mathbf{x})[\mathbf{M}(\mathbf{x})]^{-1}\mathbf{p}(\mathbf{x}_i), \qquad (1)$$

where $[\mathbf{M}(\mathbf{x})]^{-1}$ is the inverse of the moment matrix defined as

$$\mathbf{M}(\mathbf{x}) = \sum_i \omega_i(\mathbf{x},\mathbf{x}_i)\mathbf{p}(\mathbf{x_i})\mathbf{p}^T(\mathbf{x_i}).$$

A detailed account on how to construct shape functions for meshless methods can be found in [Fries and Matthies 2003]. The weight function ω_i plays an important role in this context, as it defines the support of the shape functions and thus the coupling between the nodes. We use the compactly supported radial spline function

$$\omega_i(\mathbf{x},\mathbf{y}) = \omega_i(r) = \begin{cases} 1-6r^2+8r^3-3r^4 & r \le 1 \\ 0 & r > 1, \end{cases}$$

where $r = (\|\mathbf{x}-\mathbf{y}\|)/h_i$. The support radius h_i is determined adaptively depending on the local density of nodes, which is important

when dynamically re-sampling the simulation domain during fracturing (see Section 4). Since we are mainly interested in the gradient $\nabla \mathbf{u}_i = [\nabla u_i \ \nabla v_i \ \nabla w_i]$ at the nodes $\mathbf{x}_i$, we directly apply the MLS procedure to the derivatives [Müller et al. 2004]. This yields

$$\nabla u_i = [\mathbf{A}(\mathbf{x}_i)]^{-1} \sum_j \omega_i(\mathbf{x}_i, \mathbf{x}_j)(u_j - u_i)(\mathbf{x}_j - \mathbf{x}_i), \qquad (2)$$

where $\mathbf{A}(\mathbf{x}_i) = \sum_j (\mathbf{x}_j - \mathbf{x}_i)(\mathbf{x}_j - \mathbf{x}_i)^T \omega_i(\mathbf{x}_i, \mathbf{x}_j)$ is also used for the computation of ∇v_i and ∇w_i. This approximation avoids the costly computation of the derivative of the inverted moment matrix, which would be necessary when directly computing the derivatives of Equation 1. To obtain a first order accurate approximation scheme, we use the linear basis $\mathbf{p} = [1 \ \mathbf{x}]^T$. Note that even though the weight functions are radially symmetric and given in analytical form, the shape functions and the approximation of their derivatives can in general only be evaluated numerically using Equations 1 and 2. Compared to FEM, meshless Galerkin methods typically require a higher number of integration points when solving the PDE in the weak formulation [Belytschko et al. 1994]. We overcome this problem by applying a point collocation scheme, i.e., the unknowns are only evaluated at the discrete nodes $\mathbf{x}_i$ [Zhang et al. 2001]. We also incorporate the strain state variables of [O'Brien et al. 2002] to handle plastic deformation as discussed in [Müller et al. 2004].

2.1 Modeling Discontinuities

When fracturing the model, we need to adapt the shape functions to respect the discontinuity created by a crack. Belytschko et al. [1994] introduced a visibility criterion, where nodes can only interact with each other, if the ray connecting the two node centers does not intersect a boundary surface. However, considering only this line-of-sight constraint causes undesirable discontinuities of the shape functions not only across the crack, but also within the domain (see Figure 2 (a)). We use the transparency method proposed by Organ et al. [1996] to allow partial interaction of nodes in the vicinity of the crack front. Suppose the ray between two nodes $\mathbf{x}_i$ and $\mathbf{x}_j$ intersects a crack surface at a point $\mathbf{x}_s$ (Figure 2 (c)). Then the weight function ω_i (and similarly for ω_j) is adapted to $\omega_i'(\mathbf{x}_i, \mathbf{x}_j) = \omega_i(\|\mathbf{x}_i - \mathbf{x}_j\|/h_i + (2d_s/(\kappa h_i))^2)$, where d_s is the distance between $\mathbf{x}_s$ and the closest point on the crack front, and κ controls the opacity of the crack surfaces (we use $\kappa = 1$ in all our computations). Effectively, a crack passing between two nodes lengthens the interaction distance of the nodes until eventually, in this adapted distance metric, the nodes will be too far apart to interact. As shown in Figure 2 (b) this method avoids the discontinuities of the shape functions within the domain.

3 Crack Propagation and Surface Sampling

Introducing cuts into the model exposes interior parts of the solid that need to be bounded by new surface sheets. Previous approaches based on FEM define fracture surfaces using faces of the tetrahedral elements [O'Brien and Hodgins 1999], [Bielser et al. 2003], [Müller and Gross 2004], which requires complex dynamic remeshing to avoid unnaturally coarse crack surfaces. The virtual node algorithm of [Molino et al. 2004] extends this concept by combining element boundaries with piecewise linear surface parts embedded in duplicated tetrahedra. To simplify the topological complexity and avoid stability problems during the simulation, all of these mesh-based approaches impose restrictions on where and how the material can fracture. Depending on the discretization of the domain, these constraints can lead to temporal or spatial aliasing artifacts, such as button-popping or jagged fracture lines, that degrade the realism of the animation. Our goal is to lift these restrictions and

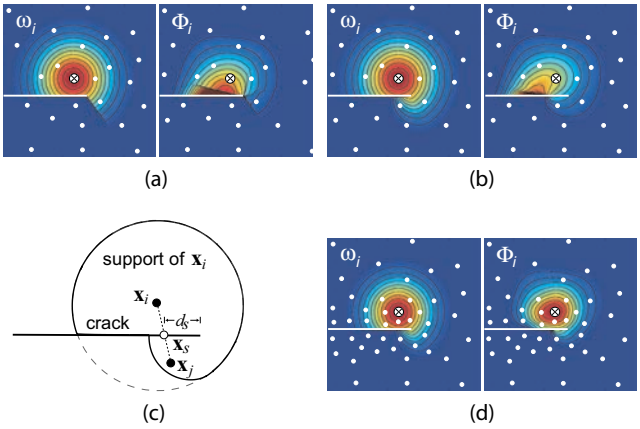

Figure 2: Comparison of visibility criterion (a) and transparency method (b) for an irregularly sampled 2D domain. The effect of a crack, indicated by the horizontal white line, on weight function ω_i and shape function Φ_i is depicted for the node $\mathbf{x}_i$ marked by the cross. A schematic view of the transparency method is shown in (c) and the effect of dynamic upsampling is illustrated in (d).

define a fracture algorithm that allows arbitrary crack initiation and propagation, while avoiding a strong dependence on the underlying discretization. To this end we use the idea of an embedded surface, but explicitly create new fracture surface sheets whenever the material is cut. The dynamic adaptation of the shape functions using the transparency method described above will then automatically adjust the simulation to the newly created fracture surfaces.

3.1 Surface Model

We use the point-based representation proposed in [Pauly et al. 2003] to model the initial object surface and all dynamically created fracture surfaces. The boundary of a 3D solid is defined by a set of surface sheets, represented as collections of overlapping, elliptical splats, called *surfels*. Since no explicit connectivity information needs to be maintained between surfels, dynamic surface sampling is simple and efficient, which is crucial for complex fracture simulations. Sharp creases and corners are represented implicitly as the intersection of adjacent surface sheets using the CSG method proposed by [Wicke et al. 2004]. The precise location of crease lines is evaluated at render time (see Figure 9), avoiding costly surface-surface intersection calculations during simulation.

Crack Model. A crack consists of a crack front and two separate surface sheets that are connected at the front to form a sharp crease. The crack front itself is defined by a linear sequence of crack nodes $\mathbf{c}_1, \ldots, \mathbf{c}_n$ that continuously add surfels to the fracture surfaces while propagating through the material. For surface cracks the end nodes of the front lie on a boundary surface or a fracture surface of a different crack. Interior cracks have circularly connected crack fronts, i.e., the two end nodes $\mathbf{c}_1$ and $\mathbf{c}_n$ coincide (see Figures 3 and 5).

3.2 Crack Initiation and Propagation

Crack initiation is based on the stress tensor σ. A new crack is created where the maximal eigenvalue of σ exceeds the threshold for tensile fracture (opening mode fracture [Anderson 1995]). This condition is evaluated for all simulation nodes. To allow crack initiation anywhere on the surface or in the interior of the model, we also apply a stochastic scheme to initiate crack fronts. We create a random set of surface and interior sample points and evaluate the

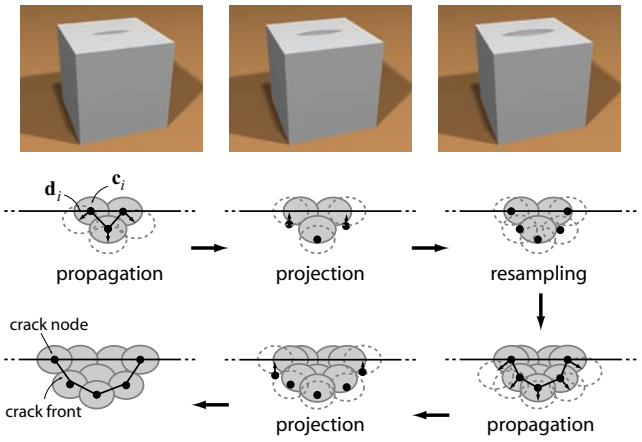

Figure 3: Front propagation and fracture surface sampling. The upper row shows a top view of an opening crack, the lower part shows a side view of a single fracture surface. After propagating the crack nodes $\mathbf{c}_i$ according to $\mathbf{d}_i$, end nodes are projected onto the surface. If necessary, the front is re-sampled and new surfels are added to the fracture surface sheets.

stress tensor at these points using weighted averaging from adjacent simulation nodes. The inherent smoothing is usually desired to improve the stability of the crack propagation [Belytschko et al. 2003]. If a crack front is initiated at one of these spatial locations, we increase the fracture thresholds of all neighboring samples to avoid spurious branching [Molino et al. 2004]. We initialize a new crack with three crack nodes, each of which carries two surfels with identical position and radius, but opposing normals. These surfels form the initial crack surfaces that will grow dynamically as the crack propagates through the solid (Figure 3). Crack propagation is determined by the propagation vectors $\mathbf{d}_i = \alpha_i \lambda_i (\mathbf{v}_i \times \mathbf{t}_i)$, where λ_i is the maximal eigenvalue of the stress tensor at $\mathbf{c}_i$, and $\mathbf{v}_i$ is the corresponding eigenvector. The vector $\mathbf{t}_i$ approximates the tangent of the crack front as $\mathbf{t}_i = (\mathbf{c}_{i+1} - \mathbf{c}_{i-1})/\|\mathbf{c}_{i+1} - \mathbf{c}_{i-1}\|$, where we set $\mathbf{c}_0 = \mathbf{c}_1$ and $\mathbf{c}_{n+1} = \mathbf{c}_n$ for surface cracks. The parameter α_i depends on the material and can be used to control the speed of propagation. The new position of a crack node $\mathbf{c}_i$ at time $t + \Delta t$ is then computed as $\mathbf{c}_i + \Delta t \mathbf{d}_i$, where Δt is the simulation time step. We additionally project the end nodes of surface cracks back onto the surface that they originated from using the projection method of [Alexa and Adamson 2004]. Since propagation alters the spacing of crack nodes along the front, we dynamically adjust the sampling resolution of the crack nodes after each propagation step. If two adjacent crack nodes are further apart than the radius of their associated surfels, a new node is inserted using cubic spline interpolation to determine the new node's position. We remove redundant crack nodes when the distance to the immediate neighbors becomes too small. Fracture surface sheets are sampled by inserting new surfels if the propagation distance exceeds the surfel radius, indicating that a hole would appear in the surface. This spatially (along the crack front) and temporally (along the propagation vectors) adaptive sampling scheme ensures uniformly sampled and hole-free crack surfaces (see Figure 3).

Transparency Weights. We determine the transparency weight $\omega_i'(\mathbf{x}_i, \mathbf{x}_j)$ for a pair of simulation nodes by computing the intersection point on the fracture surface of the ray connecting the two nodes (Section 2.1) using the method proposed by [Alexa and Adamson 2004]. The distance d_s to the crack front is approximated as the shortest Euclidean distance to the line segments defined by adjacent crack nodes.

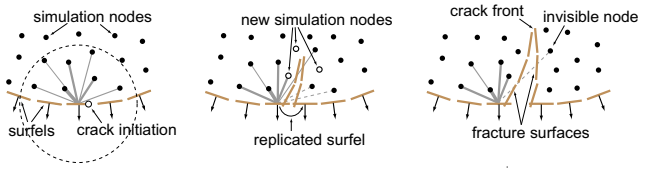

Figure 4: Transparency weights for embedding surfels in the simulation domain. The thickness of the lines indicates the influence of a simulation node on the displacement of a surfel. During crack propagation, new surfels and simulation nodes are created using dynamic re-sampling as described below.

To animate the boundary surface of the solid, we apply a free-form deformation approach that embeds surface sheets in the volumetric domain. The displacements of surfels are computed as a weighted average from neighboring simulation nodes, using the gradient approximation $\nabla \mathbf{u}_i$ as described in [Müller et al. 2004]. Here the use of the transparency method (Section 2.1) is crucial to compute these weights, as it ensures a smooth displacement field at the crack front (see Figure 4). We exploit the fact that during simulation, changes of the transparency weights are localized to a small region around the crack front. Thus only a small fraction of the weights need to be updated in every time step, leading to an efficient implementation.

3.3 Topology Control

The major challenge when explicitly modeling fracture surfaces is the efficient handling of all events that affect the topology of the boundary surfaces and the simulation domain. Apart from crack initiation, we have identified three fundamental events that are sufficient to describe the often intricate constellations that occur during fracturing: *Termination*, *splitting*, and *merging* of crack fronts:

- A crack is terminated if the crack front has contracted to a single point.
- Splitting occurs when a crack front penetrates through a surface as shown in Figure 5 (a). We use the method proposed in [Alexa and Adamson 2004] to estimate the signed distance of a crack node to a surface sheet and initiate a splitting event when a sign change occurs from one time step to the next. The front is split at the edges that intersect the surface, discarding all nodes that are outside the solid, except the ones that are connected to an interior node. These nodes become new end nodes by moving them to the intersection point with the surface. As shown on the left of Figure 5 (a), a surface crack is split into two new crack fronts that share the same crack surfaces, i.e. independently add surfels to the same fracture surface sheets during propagation. An interior crack becomes a surface crack after splitting, as illustrated on the right.
- A merging event is triggered when two surface end nodes of two crack fronts meet by creating the appropriate edge connections (Figure 5 (b)). Two surface cracks are merged into a single surface crack (left), while a circular front is created if the two end nodes are from the same crack front (right). Typically, when cracks merge, their fracture surfaces create a sharp corner, so we maintain separate fracture surface sheets that intersect to create a crease.

As can be seen in Figure 5, splitting and merging are dual to each other. The former introduces two new end nodes, while the latter decreases the number of end nodes by two. Similarly, crack initiation and termination are dual topological operations. Note that the

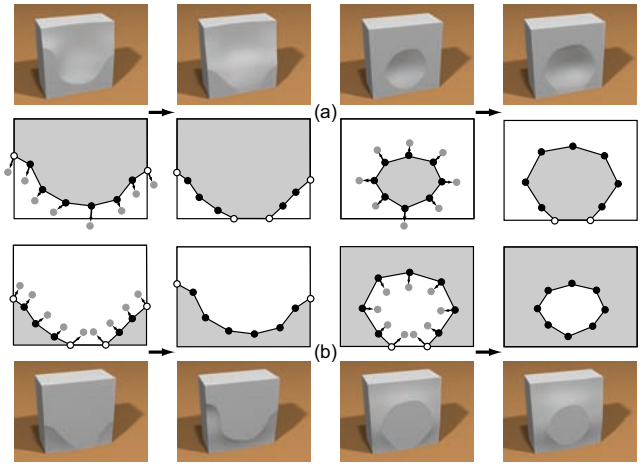

Figure 5: Topological events during crack propagation. (a) Splitting, (b) merging. The top and bottom rows show a cutaway view with one crack surface exposed. The sketches in the center rows show this fracture surface in gray, end nodes of crack fronts are indicated by white dots.

intersection of two crack fronts at interior nodes is handled automatically by first splitting both fronts and then merging the newly created end nodes.

Snapping. One technique that we found particularly useful to improve the stability of the simulation is *snapping*. Snapping guarantees that problematic small features, such as tiny fragments or thin slivers, do not arise. It works by forcing nodes very near other nodes or very near surfaces to become coincident to ensure that any features present are of size comparable to the local node spacing. Similar methods have been proven to guarantee topological consistency with the ideal geometry in other settings [Guibas and Marimont 1995]. Specifically, when a front intersects a surface, we project all crack nodes that are within snapping distance d to the surface onto the surface. This avoids fragmenting the front into small pieces that would be terminated anyway within a few time steps. We merge fronts when the end nodes are within distance d by moving both end nodes to their average position. This avoids small slivers of material to be created, which would require a significant number of new simulation nodes to be added to the model (see Section 4). Similarly, the intersection of two crack fronts can lead to multiple splitting and merging events, which we combine into a single event to avoid the overhead of creating and subsequently deleting many small crack fronts. We also apply snapping to front termination, where we delete a crack front when all its nodes are within distance d from each other. We found the average local surfel radius to be a good choice for d, as it relates directly to the sampling resolution of the model surface.

4 Volumetric Sampling

One of the main advantages of meshless methods lies in the fact that they support simple and efficient sampling schemes. To simulate the dynamics of a 3D object defined by a boundary surface S, we first need to discretize the volume V bounded by S, i.e., compute the set of simulation nodes $\{\mathbf{x}_i\}$. Similar to adaptive finite element meshing, we want a higher node density close to the boundary surface and fewer nodes towards the interior of the solid. We compute the nodal sampling of V using a balanced octree hierarchy as shown in Figure 6. Starting from the bounding box of S, a cell of the octree

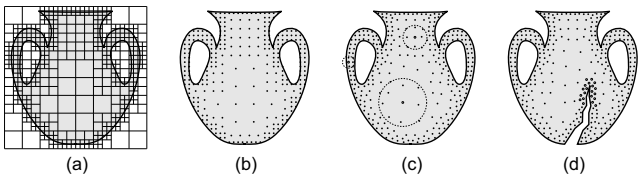

Figure 6: Volumetric sampling: (a) octree decomposition, (b) initial adaptive octree sampling, (c) sampling after local repulsion, where circles indicate 0.1 iso-value of weight function, (d) dynamic re-sampling during fracturing.

is recursively refined, if it contains parts of S. The final number of nodes is controlled by prescribing a maximum octree level at which the recursive refinement is stopped. Given this adaptive decomposition, we create a sample point at each octree cell center that lies within V. To create a locally more uniform distribution, samples are displaced within their octree cell by applying a few iterations of point repulsion. We set the support radius h_i of each node $\mathbf{x}_i$ to twice the maximum distance from $\mathbf{x}_i$ to the samples in all adjacent cells. This guarantees sufficient overlap of the nodal domains to allow stable computation of the inverted moment matrix (see Section 2). Similar to [Müller et al. 2004], each node is assigned a fixed mass $m_i = 4/3\pi h_i^3 \rho$, where ρ is the material density.

During simulation, we need to dynamically adjust the discretization of the simulation domain. Without dynamic re-sampling, frequent fracturing would quickly degrade the numerical stability of the simulation even for an initially adequately sampled model. New nodes need to be inserted in the vicinity of the crack surfaces and in particular around the crack front. At the same time, strong deformations of the model can lead to a poor spatial discretization of the simulation volume, which also requires a dynamic adaptation of the sampling resolution. This is particularly important for highly plastic materials, where the deformed shape can deviate significantly from its original configuration.

We use a simple local criterion to determine under-sampling at a node $\mathbf{x}_i$. Let $\Omega_i = \sum_j \omega_i'(\mathbf{x}_i, \mathbf{x}_j)/\omega_i(\mathbf{x}_i, \mathbf{x}_j)$ be the normalized sum of transparency weights (see Section 2.1). Without visibility constraints, Ω_i is simply the number of simulation nodes in the support of $\mathbf{x}_i$. During simulation Ω_i decreases, if fewer neighboring nodes are found due to strong deformations, or if the transparency weights become smaller due to a crack front passing through the solid. If Ω_i drops below a threshold $\Omega_{\min}$ (we use $\Omega_{\min} = 10$), new nodes are inserted around $\mathbf{x}_i$, as shown in Figure 7. We insert $\lceil \Omega_{\min} - \Omega_i \rceil$ new samples within the support radius of $\mathbf{x}_i$, similar to [Desbrun and Cani 1999]. The mass associated with $\mathbf{x}_i$ is distributed evenly

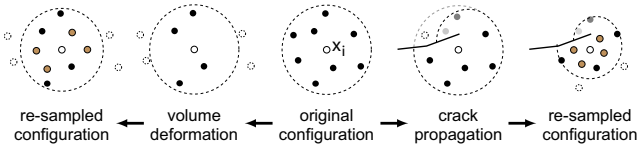

Figure 7: Dynamic re-sampling at the simulation node $\mathbf{x}_i$ due to strong deformation (left) and fracturing (right).

among the new nodes and their support radius is adapted to keep the overall material density constant. Note that mass will not be strictly preserved locally in the sense that the mass distribution of nodes after fracturing will not precisely match the correct distribution according to the separated volumes created by the fracture surface sheets. However, mass will be preserved globally and the local deviations are sufficiently small to not affect the simulation

noticeably (cf. [Molino et al. 2004]).

To prevent excessive re-sampling for nodes very close to a fracture boundary we restrict node splitting by prescribing a minimal node support radius. Note that re-sampling due to fracturing is triggered by the crack nodes passing through the solid, similar to adapting the visibility weights (see Section 3). Performing these checks comes essentially for free, since all the required spatial queries are already carried out during visibility computation. Figures 6 (d) and 9 illustrate the dynamic adaptation of the sampling rates when fracturing. The effect on the shape functions is shown in Figure 2 (d).

5 Implementation

Figure 8 shows a high-level overview of our simulation pipeline. We detect and process collisions using the method proposed in [Keiser et al. 2004]. Collision detection is based on the signed distance function of the boundary surfaces [Alexa and Adamson 2004]. Interpenetrations are resolved by computing an approximate contact surface that is consistent for both models. From the contact surface we compute penalty forces similar to [O'Brien and Hodgins 1999]. After resolving collisions and contacts, strains and stresses are computed as described in Section 2. Given the distribution of stress, we initiate new crack fronts, propagate existing cracks, and adapt the spatial sampling of the fracture surfaces (Section 3). This stage is followed by the dynamic re-sampling of the simulation domain (Section 4), before we integrate the nodal forces using an explicit Leap-frog scheme to obtain the new displacements.

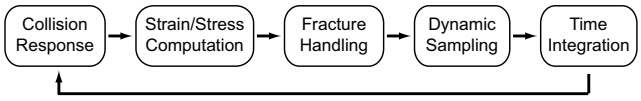

Figure 8: High-level overview of our meshless simulation pipeline.

Data Structures. For fracture simulation of elastic and moderately plastic materials, most of the computations are performed using a fixed reference system [Müller et al. 2004]. This means that the coupling of adjacent nodes can be pre-computed at the beginning of the simulation by examining the overlap of the corresponding shape functions in material coordinate space. These nodal relations are stored in a sparse neighborhood graph, where each edge e_{ij} in the graph indicates that node $\mathbf{x}_i$ interacts with node $\mathbf{x}_j$. During simulation the graph is then used to accelerate neighborhood queries. To update the neighborhood relations, we need to check if a passing crack front affects the visibility of neighboring nodes in the vicinity of the front. This is a very localized search that only requires a few ray-surface intersections and can thus be performed very efficiently. Note that this graph data structure differs greatly from a FEM mesh, where edge-, face-, and element-relations need to be kept consistent at all times. For highly plastic materials, neighborhood information is computed dynamically, using the spatial hashing grid proposed in [Teschner et al. 2003]. This data structure is also used to accelerate the collision queries, using an axis aligned bounding box hierarchy as discussed in [Keiser et al. 2004].

Rendering. All images were created using the open-source renderer POV-Ray (http://www.povray.org), which we extended to handle ray intersections with surfels as proposed in [Alexa and Adamson 2004]. As mentioned above, we avoid an explicit representation of the intersection curve of two adjacent surface sheets by deferring the surface-surface intersection problem to the rendering stage, where it can be solved efficiently. We adapt the CSG rendering technique for point-sampled surfaces proposed by Wicke et al. [2004]. During fracturing, we maintain a list of all intersecting surface sheets that form a sharp crease. With this minimal topological structure all surface-surface intersections can be resolved by the renderer during ray-casting (see Figure 9).

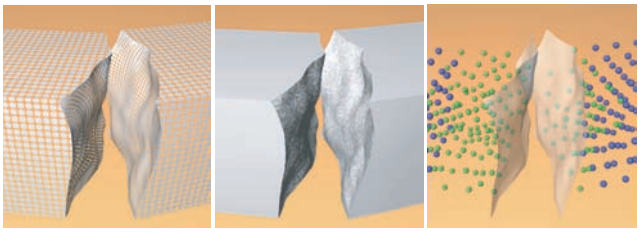

Figure 9: Surfels are clipped to create sharp creases with dynamically created fracture surfaces, whose visual roughness is controlled using 3D noise functions for bump mapping. The sampling of the simulation domain is shown on the right, where green spheres denote re-sampled nodes.

Control. Apart from specifying material properties to influence the course of the simulation, we provide explicit mechanisms for controlling fracture behavior. This is crucial in production environments and interactive applications, where the visual effect is often more important than physical accuracy. We exploit the explicit modeling of fracture surfaces and the dynamic adaptation of shape functions and nodal sampling resolution to support precise control of where and how a model fractures. To this end we have implemented a simple painting interface that allows fast prototyping of fracture simulations by prescribing fracture patterns directly on the object boundary. The user can paint arbitrary networks of cracks on the surface and explicitly specify stress thresholds for these cracks. Additionally, we provide a propagation history to control the propagation of cracks through the material. The adjusted propagation vector at time t is computed as the weighted average $\bar{\mathbf{d}}_i^t = \gamma \mathbf{d}_i^{t-\Delta t} + (1-\gamma)\mathbf{d}_i^t$, where $\gamma \in [0,1]$ is the history factor. A purely stress-based propagation is achieved for $\gamma = 0$, while $\gamma = 1$ yields purely geometric cracks and fracture surfaces. We also use volumetric textures for adjusting the fracture thresholds within the material. The pre-scoring technique of [Molino et al. 2004], where the stress tensor is modified according to an embedded level set function, can also be integrated easily.

6 Results and Discussion

Figure 1 shows brittle fracture of a stiff elastic object, computed at an average of 22 seconds per frame. The initial model is sampled with 4.3k simulation nodes and 249k surface samples. During fracturing the number of nodes increases to 6.5k, while 61k additional surfels have been created to define the new fracture surfaces. Compared to FEM-based approaches that use the faces of simulation elements to define the object surface, we achieve a significantly higher level of surface detail without requiring a proportionally large number of simulation nodes. This de-coupling of the simulation domain from the representation of the boundary surface leads to increased performance and provides essential control in computer animation, where visual quality is typically favored over physical accuracy. On the other end of the spectrum of material properties that our method can handle is the example of Figure 10. The highly plastic bubble-gum is deformed beyond recognition before splitting along a complex fracture surface. Figure 11 illustrates crack front merging events. Four of a total of 49 crack fronts merge in the center

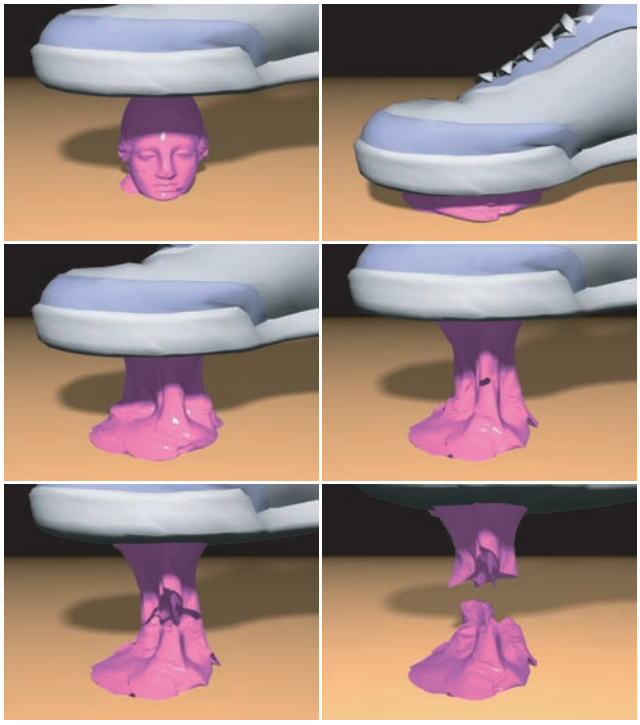

Figure 10: Highly plastic deformations and ductile fracture. Initial/final sampling: 2.2k/3.3k simulation nodes, 134k/144k surfels, 2.4 sec/frame.

of the twisted bar to form a circular crack front. Figure 12 shows how we can explicitly control fracture, using a combination of crack painting, propagation history, and adaptive fracture thresholds.

Note that all models shown in this paper have been re-sampled substantially during the simulation to match the increased geometric and topological complexity after fracturing. The simplicity of this dynamic re-sampling of the simulation domain highlights one of the main benefits of meshless methods for physics-based animation. Due to minimal consistency constraints between neighboring nodes, dynamic re-sampling is efficient and easy to implement, as compared to the far more involved re-meshing methods used in FEM simulations. A similar argument holds for our surface sampling method. Instead of maintaining a consistent surface mesh and dynamically cutting and re-meshing during simulation, our sampling scheme simply inserts and moves surfels during crack propagation. Surface-surface intersections are resolved at render time, avoiding costly topological updates during simulation. In principal, however, any surface representation that supports inside/outside queries and ray intersections can be used in our system.

Another advantage of our meshless approach is the flexibility in handling a wide range of different material properties, as shown in the examples of Figures 1 and 10. However, extremely stiff objects require very small time steps to obtain an accurate distribution of stress within the material. While this problem can be alleviated using implicit or semi-implicit integration schemes, simpler approaches such as [Smith et al. 2000] might be more appropriate for materials that do not exhibit noticeable deformations before fracturing. Coupling of deformable bodies with rigid bodies such as the ground plane in Figure 10 requires enforcement of essential boundary conditions. Currently we resolve all collisions using penalty forces on the simulation nodes, which does not guarantee that the simulation is free of self-intersections at all times. More

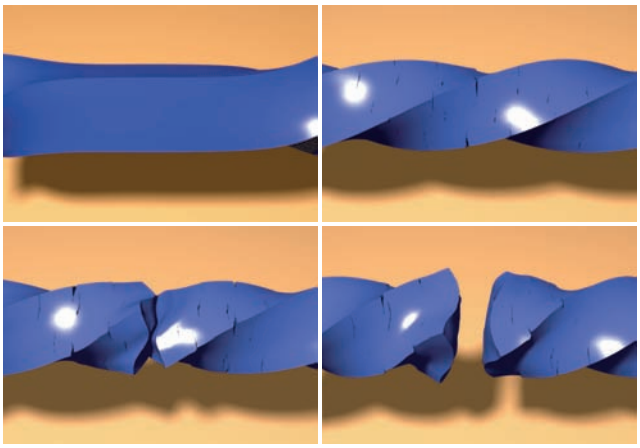

Figure 11: Crack merging. Initial/final sampling: 2k/3k simulation nodes, 29k/45k surfels, 10 sec/frame.

advanced methods such as Lagrange multipliers or the Nitsche method [Fernandez-Mendez and Huerta 2004] could be integrated into our system, but would lead to higher computational costs.

A general limitation of the meshless approach is that we require even very small fragments to be sampled sufficiently dense in order to obtain a stable evaluation of the shape functions. This inflates the number of simulation nodes when an object is fractured excessively, which slows down the computations. If performance is crucial, we resort to modeling small pieces of material as rigid bodies, assuming that the internal deformations are negligible. Fortunately, large numbers of very small fragments are mainly created by stiff objects that experience brittle fracture, where such an approximation is reasonable.

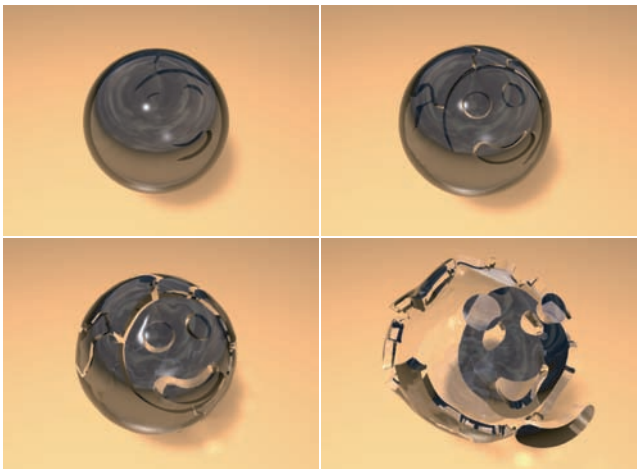

Figure 12: Controlled fracture. Initial/final sampling: 4.6k/5.8k simulation nodes, 49k/72k surfels, 6 sec/frame.

7 Conclusion and Future Work

We have introduced a new system for animating deformable objects that fracture. Instead of maintaining a consistent volumetric mesh using continuous cutting and re-structuring of finite elements, we dynamically adjust nodal shape functions based on simple visibility constraints. The space discretization is continuously adapted

using insertions of simulation nodes. Similarly, a point-based representation is built for the boundary surface, which allows efficient dynamic sampling of fracture surfaces, and facilitates explicit control of the object topology.

There are a number of interesting directions for future research. Extending our method to a hierarchical scheme would allow more efficient simulation of physical phenomena that occur at different scales. We also believe that new data structures can improve the tradeoff between local caching and dynamic re-computation, which is at the core of our meshless animation system. In addition we plan to extend our physical model to include more complex constitutive models, allow the modeling of thin shells, and support coupling between deformable bodies and fluids.

Acknowledgements. The authors wish to acknowledge support from NSF grants CARGO-0138456, ITR-0205671, ARO grant DAAD19-03-1-033, and NIH Simbios Center grant 1091129-1-PABAE. Bart Adams is funded as a Research Assistant by the Fund for Scientific Research - Flanders, Belgium (F.W.O.-Vlaanderen). Thanks to Eitan Grinspun and the anonymous reviewers for helpful comments.

References

ALEXA, M., AND ADAMSON, A. 2004. On normals and projection operators for surfaces defined by point sets. In *Proceedings of Eurographics Symposium on Point-Based Graphics*, 150–155.

ANDERSON, T. L. 1995. *Fracture Mechanics*. CRC Press.

BELYTSCHKO, T., LU, Y., AND GU, L. 1994. Element-free galerkin methods. *Int. J. Numer. Meth. Engng 37*, 229–256.

BELYTSCHKO, T., KRONGAUZ, Y., ORGAN, D., FLEMING, M., AND KRYSL, P. 1996. Meshless methods: An overview and recent developments. *Comp. Meth. in Appl. Mech. Eng. 139*, 3.

BELYTSCHKO, T., CHEN, H., XU, J., AND ZI, G. 2003. Dynamic crack propagation based on loss of hyperbolicity and a new discontinuous enrichment. *Int. J. Numer. Meth. Engng 58*, 1873–1905.

BIELSER, D., GLARDON, P., TESCHNER, M., AND GROSS, M. 2003. A state machine for real-time cutting of tetrahedral meshes. In *Pacific Graphics*, 377–386.

CARLSON, M., MUCHA, P., VAN HORN III, B., AND TURK, G. 2002. Melting and flowing. In *Proceedings of the 2002 ACM SIGGRAPH Symposium on Computer Animation*.

CHANG, J., AND ZHANG, J. J. 2004. Mesh-free deformations. In *Comp. Anim. Virtual Worlds*, 211–218.

CHUNG, T. J. 1996. *Applied Continuum Mechanics*. Cambridge Univ. Press, NY.

DEBUNNE, G., DESBRUN, M., CANI, M.-P., AND BARR, A. 2001. Dynamic real-time deformations using space & time adaptive sampling. In *Computer Graphics Proceedings*, Annual Conference Series, ACM SIGGRAPH 2001, 31–36.

DESBRUN, M., AND CANI, M.-P. 1995. Animating soft substances with implicit surfaces. In *Computer Graphics Proceedings*, ACM SIGGRAPH, 287–290.

DESBRUN, M., AND CANI, M.-P. 1996. Smoothed particles: A new paradigm for animating highly deformable bodies. In *6th Eurographics Workshop on Computer Animation and Simulation '96*, 61–76.

DESBRUN, M., AND CANI, M.-P. 1999. Space-time adaptive simulation of highly deformable substances. Tech. rep., INRIA Nr. 3829.

FEDKIW, R., STAM, J., AND JENSEN, H. W. 2001. Visual simulation of smoke. In *SIGGRAPH 2001, Computer Graphics Proceedings*, ACM Press / ACM SIGGRAPH, E. Fiume, Ed., 15–22.

FELDMAN, B. E., O'BRIEN, J. F., AND ARIKAN, O. 2003. Animating suspended particle explosions. In *Proceedings of ACM SIGGRAPH 2003*, 708–715.

FERNANDEZ-MENDEZ, S., AND HUERTA, A. 2004. Imposing essential boundary conditions in mesh-free methods. *Computer Methods in Applied Mechanics and Engineering 193*, 1257–1275.

FRIES, T.-P., AND MATTHIES, H. G. 2003. Classification and overview of meshfree methods. Tech. rep., TU Brunswick, Germany Nr. 2003-03.

GUIBAS, L. J., AND MARIMONT, D. H. 1995. Rounding arrangements dynamically. In *SCG '95: Proceedings of the eleventh annual symposium on Computational geometry*, ACM Press, 190–199.

HIROTA, K., TANOUE, Y., AND KANEKO, T. 1998. Generation of crack patterns with a physical model. *Vis. Comput. 14*, 126–137.

JAMES, D. L., AND PAI, D. K. 1999. Artdefo, accurate real time deformable objects. In *Computer Graphics Proceedings*, Annual Conference Series, ACM SIGGRAPH 99, 65–72.

KEISER, R., MÜLLER, M., HEIDELBERGER, B., TESCHNER, M., AND GROSS, M. 2004. Contact handling for deformable point-based objects. In *Proceedings of VMV*.

KRYSL, P., AND BELYTSCHKO, T. 1999. The element free galerkin method for dynamic propagation of arbitrary 3-d cracks. *Int. J. Numer. Meth. Engng 44*, 767–800.

LANCASTER, P., AND SALKAUSKAS, K. 1981. Surfaces generated by moving least squares methods. *Mathematics of Computation 87*, 141–158.

LIU, G. R. 2002. *Mesh-Free Methods*. CRC Press.

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

MÜLLER, M., AND GROSS, M. 2004. Interactive virtual materials. In *Proceedings of the 2004 conference on Graphics interface*, Canadian Human-Computer Communications Society, 239–246.

MÜLLER, M., MCMILLAN, L., DORSEY, J., AND JAGNOW, R. 2001. Real-time simulation of deformation and fracture of stiff materials. *EUROGRAPHICS 2001 Computer Animation and Simulation Workshop*, 27–34.

MÜLLER, M., KEISER, R., NEALEN, A., PAULY, M., GROSS, M., AND ALEXA, M. 2004. Point based animation of elastic, plastic and melting objects. *Proceedings of 2004 ACM SIGGRAPH Symposium on Computer Animation*, 141–151.

NGUYEN, D. Q., FEDKIW, R., AND JENSEN, H. W. 2002. Physically based modeling and animation of fire. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, 721–728.

O'BRIEN, J. F., AND HODGINS, J. K. 1999. Graphical modeling and animation of brittle fracture. In *Proceedings of SIGGRAPH 1999*, 287–296.

O'BRIEN, J. F., BARGTEIL, A. W., AND HODGINS, J. K. 2002. Graphical modeling and animation of ductile fracture. In *Proceedings of SIGGRAPH 2002*, 291–294.

ORGAN, D., FLEMING, M., TERRY, T., AND BELYTSCHKO, T. 1996. Continuous meshless approximations for nonconvex bodies by diffraction and transparency. *Computational Mechanics 18*, 1–11.

ORTIZ, M., AND PANDOLFI, A. 1999. Finite-deformation irreversible cohesive elements for three-dimensional crack-propagation analysis. *Int. J. Num. Meth. Eng. 44*, 1267–1282.

PAULY, M., KEISER, R., KOBBELT, L. P., AND GROSS, M. 2003. Shape modeling with point-sampled geometry. *ACM Trans. Graph. 22*, 3, 641–650.

SMITH, J., WITKIN, A., AND BARAFF, D. 2000. Fast and controllable simulation of the shattering of brittle objects. In *Graphics Interface*, 27–34.

SUKUMAR, N., MOS, N., MORAN, B., AND BELYTSCHKO, T. 2000. Extended finite element method for three-dimensional crack modeling. *International Journal for Numerical Methods in Engineering 48*, 11, 1549–1570.

TERZOPOULOS, D., AND FLEISCHER, K. 1988. Modeling inelastic deformation: viscolelasticity, plasticity, fracture. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, ACM Press, 269–278.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. In *Computer Graphics Proceedings*, Annual Conference Series, ACM SIGGRAPH 87, 205–214.

TESCHNER, M., HEIDELBERGER, B., MÜLLER, M., POMERANETS, D., AND GROSS, M. 2003. Optimized spatial hashing for collision detection of deformable objects. In *Proc. Vision, Modeling, Visualization VMV*, 47–54.

VENTURA, G., XU, J., AND BELYTSCHKO, T. 2002. A vector level set method and new discontinuity approximations for crack growth by efg. *International Journal for Numerical Methods in Engineering 54*, 923–944.

WICKE, M., TESCHNER, M., AND GROSS, M. 2004. Csg tree rendering of point-sampled objects. In *Proceedings of Pacific Graphics 2004*.

ZHANG, X., LIU, X.-H., SONG, K.-Z., AND LU, M.-W. 2001. Least-squares collocation meshless method. *Int. J. Numer. Meth. Engng 51*, 1089–1100.