Eurographics Symposium on Point-Based Graphics (2006)
M. Botsch, B. Chen (Editors)

Integrating Mesh and Meshfree Methods for Physics-Based Fracture and Debris Cloud Simulation

Nan Zhang, Xiangmin Zhou, Desong Sha, Xiaoru Yuan, Kumar Tamma, and Baoquan Chen

University of Minnesota at Twin Cities, USA

Abstract

We present a hybrid framework for physics-based simulation of fracture and debris clouds. Previous methods mainly consider bulk fractures. However, in many situations, small fractured pieces and debris are visually important. Our framework takes a hybrid approach that integrates both tetrahedron-based finite element and particle-based meshfree methods. The simulation starts with a tetrahedral mesh. When the damage of elements reaches a damage failure threshold, the associated nodes are converted into mass-based particles. Molecular dynamics is used to model particle motion and interaction with other particles and the remaining elements. In rendering, we propose an algorithm of dynamically extracting a polygonal boundary surface for the damaged elements and particles. Our framework is simple, accurate, and efficient. It avoids the remeshing and stability problems of pure mesh-based techniques and pure meshfree methods and offers high visual realism.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physics based modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

Realistic animation of material fracture and the subsequent fractured debris clouds remains a challenging problem in the Computer Graphics community. Prominent existing methods [MBF04, OH99, PKA*05] for physics-based fracture simulation mainly consider bulk fractures. When considering long term total effect of the material damage/fracture/penetration animation, the failed material is one of the crucial parts of the modeling. This is because the interaction between the failed material and the unfractured bulk material will cause further fracture of the bulk material and dictate the post initial fracture behavior of the complete animation. Current fracture simulation approaches are unable to model the fracture debris and the behavior of the failed material accurately and efficiently.

In this paper we propose a hybrid solution to the animation of the fracture behavior of multiple body interactions. Our approach initially models the entire interaction system with the finite element method. When the continuum damage of the elements reaches a critical state, the nodes of the damaged elements are converted into meshless particles and subjected to the associated pressure fields. The particles are modeled by the molecular dynamics (MD) method to account for the interaction between the particles, and the interaction with the unfractured elements. This generalized framework for animating the fracture behavior of multiple objects and interactions is a composite of individual ingredients of the state-of-the-art development of the respective research fields. The combination of both mesh and meshfree techniques makes this approach a hybrid simulation method. Figure 1 shows such a simulation result computed by our simulation framework.

The primary reasons for the choice of the ingredients in our approach are: FEM has been well established for modeling contact problems, but has difficulty in handling discrete fields and changing boundary. Meshfree methods are good at modeling debris clouds. In some physical events such as deep penetration, debris clouds play an important role for the visual effects. Thus, the combination of the two methods becomes crucial. Our contributions in this paper are:

- The integration of both mesh and meshfree methods, in particular, the particle based MD method with FEM. We

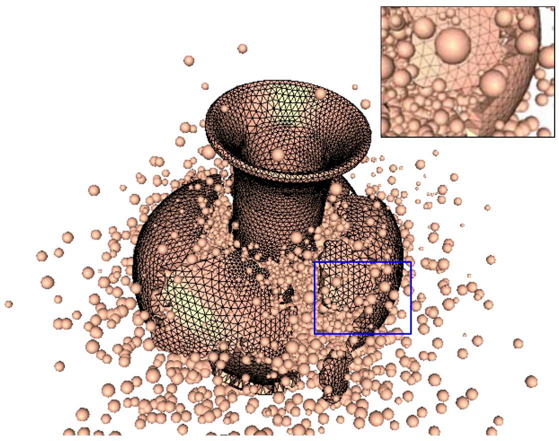

Figure 1: *A vase shattered onto the ground, where particles are used to model the fractured debris cloud. A zoomed-in view of the boxed region is shown at the top-right corner.*

accurately model the underlying physics for the animation of the fracture behavior of multiple objects and their interactions to provide a realistic scenario of the physics.

- A visualization technique to dynamically extract polygonal boundary surfaces for failed elements and particles, and a reconstruction of the physical representation of dynamic systems from the discrete simulation outputs.

This paper is organized as follows. We discuss the related work in Section 2. In Section 3 we first give an outline of the simulation framework, then present the physical laws governing the simulation. In Section 4 we propose a visualization technique to extract the polygonal boundary surfaces for the particles. Experimental results are presented in Section 5. Finally we conclude the paper in Section 6.

2. Related Work

Previous work in fracture simulation can be classified as non-physics, FEM, and meshfree methods. Non-physics based methods are popular in the early stage of fracture modeling. Neff and Fiume [NF99] model the destructed wall structure caused by a spherical blast wave using an artificial pattern generator. Hirota et al. [HTK98] use a mass-spring model to create static crack patterns. Smith et al. [SWB01] and Norton et al. [NTB*91] model dynamic cracks, such as a teapot shattered onto the ground, using a constraint based method and a mass-spring method, respectively. Although these methods allow for easy control of fracture patterns and are simple and fast, they do not provide very realistic results.

Terzopoulos and Fleischer [TF88] have pioneered physics based simulation by using finite differences to handle plastic deformation and cloth tearing. Later on, based on research from computational mechanics, researchers started to work on finite element methods that directly approximate the equations of continuum mechanics. O'Brien et al. are the first to apply this technique for simulating brittle fractures [OH99] and ductile fractures [OBH02]. To conform with the fracture lines that are derived from the principal stresses, elements are dynamically cut and remeshed. However, remeshing can be expensive and reduces the time step for simulation. To avoid these issues, Molino et al. [MBF04] propose a virtual node algorithm, where elements are duplicated and fracture surfaces are embedded in the copied tetrahedra. Building upon the recent advances in meshfree methods for computational mechanics, Pauly et al. [PKA*05] extend the meshfree surface deformation framework of Müller et al. [MKN*04] to deform solid objects that fracture. In their system, both sparse simulation nodes and dense boundary points are maintained. To adapt the simulation nodes to the fractures, new simulation nodes are inserted. Similarly, new points are generated at the crack interface. The explicit modeling of advancing crack fronts and associated fracture surfaces makes it easy to generate highly detailed, complex fracture patterns, but requires handling of topological operations on crack fronts. In our application domain, topology maintaining operations are very difficult to achieve. Note that it is also possible to model fracture surfaces implicitly in a meshfree fracturing method, such as the level set method [VXB02]. However, it is expensive to model a large amount of tiny fractures using such techniques.

Particle-based methods have been widely used in computer graphics for simulating a wide range of amorphous phenomena, such as fluid [FF01], smoke [FSJ01], and explosion [FOA03]. In these methods, particle motion is the primary concern and is computed by physical equations such as the Navier-Stokes. Particles have also been used for granular material simulation. Bell et al. [BYM05] use anisotropical particles and molecular dynamics equations to simulate splashing and avalanche of sands. They also compute interacting forces between particles and rigid objects by sampling the boundary surfaces of rigid objects into particles, but no fractures are involved. For deformable solid modeling, Desbrun and Cani [DC95] model soft inelastic materials using particle systems coated with a smooth iso-surface. In all these methods, particles are generated in the beginning, which is much different from our simulation method that generates particles from damaged elements. Another distinct difference is that in these systems particles are used to model amorphous objects and can be rendered using texture splats or isosurfaces. In our method, particles are given a clear, sharp, polygonal boundary to provide meaningful simulation of the actual fragmented pieces.

The meshfree/particle method was initially developed by the computational mechanics community. A detailed summary of this technique is beyond the scope of this paper. For some excellent reviews, please refer to [Liu02]. In hybrid particle/FEM interaction, Johnson et al. [JBS00,JBS01] propose a generalized particle algorithm (GPA) for high velocity impact and other dynamics problems, where a variation of the smooth particle hydrodynamics (SPH) method is used to model particle motion, which is different from the molecular dynamics method used in our framework.

3. Simulation

3.1. Overview

Consider an interactive system of at least two individual continuum objects with initial velocities and subjected to a gravity field. The objects are discretized in space by finite element meshes with all the surface elements treated as master contact segments and all the surface nodes treated as slave contact nodes. Contact search is performed in every time step to find the corresponding contactable master segment and slave node pairs. When impact occurs, the objects subjected to the contact forces undergo finite strain visco-elasto-plastic deformation. If the accumulated effective plastic strain of any given Gauss point of the element reaches the threshold value, the continuum damage effect needs to be accounted for the deviatoric stress. At the same time, the equation of state (EOS) is considered for the hydrostatic pressure of the material to account for the shock physics. When the damage of a Gauss point reaches a critical value, the material of the corresponding finite element is considered failed. The material of the finite element can no longer handle any shear stresses, but still retains the hydrostatic pressure. At this time, since the finite element cannot withstand any shear stresses, it mostly likely undergoes severe distortion and causes the simulation to stop. Therefore, it is removed from the continuum body. The mass and the hydrostatic pressure of the failed finite element are distributed to the corresponding nodes of the element. The boundary surfaces are regenerated as new master contact segments. The finite element nodes of the failed elements are then treated as mass based particles using the rule described in Section 3.2. Particles are modeled by the molecular dynamics method to account for the interactions amongst the mass particles and between the mass particles and the existing master contact segments. They are not divided any further.

One may suggest converting failed elements into rigid bodies and simulating them using a rigid body engine. Although it is simpler, it does not account for all possible physical equations representing the model and the underlying mechanics. The conversion of the failed elements is to continually model the pressure field resulting from the equation of state. Converting to rigid bodies or using a simple approach will not correctly model the underlying physics.

3.2. Element/Particle Conversion

In [JBS00], a straightforward strategy is given to convert a failed element into a particle. A particle inherits the mass of the element and is positioned in the element center. In our system, we remove a failed element from the volumetric mesh, too. Whether or not to generate a particle depends on its neighbors. The particle conversion rule is: when all the elements connected with a node fail, we then convert this node into a particle. The particle carries a quarter mass from all its connecting elements and inherits the node's current position. In Figure 2, we compare the two different converting strategies. Although particles should have different geometric shapes, accurately evaluating their actual geometric shapes is very expensive and does not contribute much to the correctness of the simulation, since particles are small. By treating the particles as isotropic spheres in the simulation stage, the computation cost is tremendously reduced.

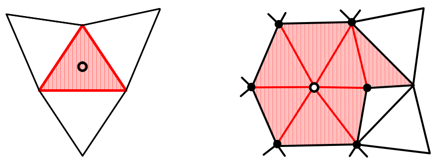

Figure 2: *Two element/particle converting strategies. (Left) One element to one particle. (Right) Our strategy, where all connecting elements must fail. Because of the strict limitation, only one particle is generated.*

The advantages of our converting strategy are: (1) Because of the restrictions we have enforced, much fewer particles are generated than in the method of [JBS00], while the element count remains the same. Therefore, the computational cost is reduced. (2) There are no new vertices generated, which eases the dynamic memory management. Many matrices do not need to be recomputed. (3) This kind of particle conversion makes the visualization technique discussed in Section 4 possible. Instead of rendering isotropic spheres, we extract a polygonal boundary for each particle to achieve more realistic visualization for the fragmented pieces.

3.3. Governing Equation: $F = ma$

The governing equation describes the force balance relation amongst the space discretized mathematical abstract material points and is essentially the Newton's second law: $F = ma$, where F is force acting on the discrete material point, m is the associated mass, and a is the acceleration. Let the open set $\Omega_t \subset \Re^3$ be the domain of interest at the configuration of time t, with the boundary $\partial\Omega_t = \Gamma_t^f \cup \Gamma_t^d$, $\Gamma_t^f \cap \Gamma_t^d = \emptyset$, and the closure $\overline{\Omega}_t = \Omega_t \cup \partial\Omega_t$. Define the set of virtual displacement field as, $[H_0^1(\Omega_t)]^3 := \{w \mid w \in [H^1(\Omega)]^3,\ w = \mathbf{0} \text{ on } \Gamma^d\}$, and the set of virtual velocity field as, $[H_v^1(\Omega_t)]^3 := \{v \mid v \in [H^1(\Omega)]^3,\ v = \dot{x}_t^d \text{ on } \Gamma^d\}$, where $H^1(\Omega)$ is the first order differentiable Hilbert space. The Eulerian weak form of the dynamic equilibrium equation subjected to the contact boundary condition is given as the following.

$$
\begin{aligned}
&(w, \frac{\partial(\rho v)}{\partial t} + \rho \eta v)_{\Omega_t} + (\mathscr{D}(w), \sigma)_{\Omega_t} \\
=\ &(w,b)_{\Omega_t} + (w,f)_{\Gamma_f(t)} + (\delta g, \sigma_N + \sigma_\tau)_{\Gamma_c(t)} \qquad (1) \\
&(\tau_N, g_N - \lambda_N^c N)_{\Gamma_c(t)} = 0 \qquad (2) \\
&(\tau_\tau, \dot{g}_\tau - \lambda_\tau^c \nabla_{\sigma_\tau} \psi_\tau(\sigma_\tau))_{\Gamma_c(t)} = 0 \qquad (3)
\end{aligned}
$$

and subject to the constraints $\lambda_N^c \geq 0$, $\psi_N(\sigma_N) = \sigma_N \cdot N \geq 0$, $\lambda_N^c \psi_n(\sigma_N) = 0$, on $\Gamma_c(t)$, and $\psi_\tau(\sigma_\tau) \geq 0$, $\lambda_\tau^c \geq 0$, $\lambda_\tau^c \psi_\tau(\sigma_\tau) = 0$, on $\Gamma_c(t)$. Where ρ is material density, v is the velocity, η is the viscous damping, σ is the Cauchy stress, b is the body force, f is the external force, g is the contact gap vector of the slave-node/master-segment pair, σ_N is the normal contact stress, σ_τ is the tangential contact stress, τ_N is the variation of the normal contact stress, τ_τ is the variation of the tangential contact stress, g_N is the normal contact gap vector, g_τ is the tangential contact gap vector, λ_N^c and λ_τ^c are the Lagrangian multipliers, $\psi_\tau(\sigma_\tau)$ is the friction law, and $(\bullet,\bullet)_\Gamma = \int_\Gamma \bullet : \bullet ds$ represents the inner-product. The solution method for the contact boundary conditions is described in [ZST03].

The mathematical modeling of the failed material or debris clouds proposed in this paper is adopted from the mathematical modeling of atomic interaction of the molecular dynamics method as described in [AW57]. For a particular mass based particle i, the governing equation is the Newton's equation of motion given by,

$$F_i = m_i \frac{d^2 x_i}{d^2 t} \qquad (4)$$

where F_i is the force exerted on the particle i, m_i is the mass, and x_i is the position of the center point of i. The force F_i is the sum of all the interaction force between particle i and the neighboring particles. And the interaction force between particle i and particle j is given by,

$$F_j = -\nabla_r U(r_{ij}) \qquad (5)$$

where $r_{ij} = \|x_i - x_j\|$ is the distance between particles i and j, and $U(r_{ij})$ is the potential energy function. Each of the particles also serves as the slave node for the contact formulation. The contact forces are also needed to account for when the gap vector of the associated slave node to the master segment is zero.

3.4. Constitutive Models

The simulation described in this exposition is modeled in conjunction with the following three aspects: (i) the Coulomb friction law for modeling the contact-impact interaction, (ii) the continuum damage mechanics based hydrodynamic equation for modeling the continuum material behavior, and (iii) the EOS induced soft-sphere potential for modeling the failure material behavior. The continuum equation is discretized using linear tetrahedral elements, since higher order elements cannot model contact physics.

3.4.1. Friction Law

The contact between the continuum bodies and the contact between the failed material and the continuum bodies are modeled by the following Coulomb friction law.

$$\psi_\tau(\sigma_\tau) = \mu^c \|\sigma_N\| - \|\sigma_\tau\| \geq 0 \quad \text{on} \quad \Gamma_c \times [0, \infty) \qquad (6)$$

where $\sigma_N \in \Re^3$ is the normal contact stress, $\sigma_\tau \in \Re^3$ is the tangential contact stress, and $\mu^c \in \Re$ is the Coulomb friction coefficient.

3.4.2. Hydrodynamic Equation

In the governing equation, the internal force vector is a result from the constitutive equation which dictates the behavior of the material. The material behavior of the continuum bodies is modeled by the continuum damage mechanics based hydrodynamic equation given as

$$
\left\{
\begin{array}{l}
\widetilde{\sigma} = \dfrac{\sigma}{1-\varphi} = \widetilde{\sigma}_D + \widetilde{\sigma}_H \\
\widetilde{\sigma}_D^{\nabla} = 2G\mathscr{D}_D^e \\
\widetilde{\sigma}_H = -p(\rho, e)I
\end{array}
\right. \qquad (7)
$$

where $\widetilde{\sigma}$ is the effective stress tensor, σ is the Cauchy stress tensor, φ is the isotropic damage state variable which is the ratio between the continuum damaged volume represented by the micro-void volume and the volume of the material (see Figure 3 for visualization), $\widetilde{\sigma}_D$ is the deviatoric portion of the effective stress tensor, $\widetilde{\sigma}_H$ is the hydrostatic portion of the effective stress tensor, G is the shear modulus, $\mathscr{D}_D^e$ is the elastic part for the deviatoric portion of the velocity strain tensor, p is the hydrostatic pressure, ρ is the material density, e is the internal energy per unit mass, I is the rank two identity tensor, and the symbol $\bullet^{\nabla}$ denotes to the Truesdell stress rate [Tru52]. The combination of the continuum damage mechanics with the hydrodynamic equation and the use of the Truesdell stress rate for the hydrodynamic equation are novel and yield meaningful and accurate physics for this application.

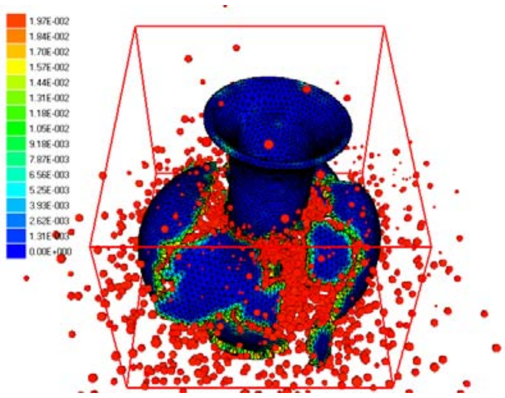

Figure 3: *Visualizing the scalar damage values of the simulation nodes using 1D texture-mapping.*

The deviatoric portion of the hydrodynamic equation incorporates the Truesdell stress rate hypo-elasto-plastic model, the Lemaitre plastic-damage model, and the Johnson-Cook fracture model together. The deviatoric portion of the hydrodynamic equation is obtained after some straight forward but involved derivation as the following.

$$\widetilde{\sigma}_D^{\nabla} = \left(\mathscr{C} - \frac{\mathscr{C} : n^{tvpd} \otimes n^{tvpd} : \mathscr{C}}{n^{tvpd} : \mathscr{C} : n^{tvpd} + H^{tvpd}} \right) : \mathscr{D}_D \tag{8}$$

where $\mathscr{C} = 2G\mathscr{I}$ is the elasticity tensor, $\mathscr{I}$ is the fourth-order identity tensor, $H^{tvpd} = h + C(h + \frac{\dot{\varepsilon}_0 R_{tvp}}{\dot{\varepsilon}\Delta t}) \ln(\frac{\dot{\varepsilon}}{\dot{\varepsilon}_0}) - \frac{H(\varepsilon - \varepsilon_D)\sigma_{eq}}{(1-\varphi)^2(\varepsilon_F - \varepsilon_D)}$, $h = Bn\varepsilon^{n-1}(1 - \tilde{T}^m)$, $n^{tvpd} = \frac{3\sigma_D}{2\sigma_{eq}}$, $\dot{\varepsilon} = \dot{\varepsilon}_0 e^{\frac{f_{rp}}{CR_{tvp}}} H(f_{rp})$, $f_{rp} = CR_{tvp} \ln(\frac{\dot{\varepsilon}}{\dot{\varepsilon}_0})$, $R_{tvp} = (A + B\varepsilon^n)(1 - \tilde{T}^m)$, $\varepsilon_F = (D_1 + D_2 e^{D_3 \frac{\sigma_H}{\sigma_{eq}}})(1 + D_4 \ln(\frac{\dot{\varepsilon}}{\dot{\varepsilon}_0}))(1 + D_5\tilde{T})$, $H(\bullet)$ is the Heaviside function, A, B, C, n, m and $D_1 - D_5$ are material constants, $\varepsilon_0 = 1.0s^{-1}$, $\tilde{T} = (T - T_{room})/(T_{melt} - T_{room})$, T_{room} is the room temperature, T_{melt} is the material melting temperature, ε_D is the uniaxial tension damage threshold strain, σ_{eq} is the von Mises equivalent stress, and $\mathscr{D}_D$ is the deviatoric part of the velocity strain. The stress update formulation can be found in [ZST06].

The temperature is governed by the temperature equation as the following.

$$\rho c \dot{T} = \eta_e \sigma : (\mathscr{D}_D - \mathscr{D}_D^e) \tag{9}$$

where ρ is the material density, c is the material heat capacity, and $\eta_e = 0.9$ is efficiency ratio [BCHW93]. Equation (9) represents that the temperature change of the system (left hand side) is equal to a portion of the dissipated energy of the system (right hand side).

The hydro static pressure is determined from the equation of state (EOS) along with the material internal energy equation [WC55]. The rate of internal energy per current unit mass is given as

$$\dot{e} = \frac{1}{\rho}(\sigma_D : \mathscr{D} - p\, tr(\mathscr{D})) \tag{10}$$

where p is the hydrostatic pressure, and $tr(\mathscr{D})$ is the trace of the velocity strain. The Mie-Grüneisen equation of state considered here is given by

$$p(\rho, e) = (1 - \frac{1}{2}\gamma\mu)p_H + \gamma\rho e \tag{11}$$

where $\mu = \eta - 1$, $\eta = \rho/\rho_0$, ρ_0 is the material density at the initial reference state, γ is the Grüneisen parameter and is given as

$$\gamma = \begin{cases} \frac{\gamma_0 \rho_0}{\rho} & \eta > 1 \\ \gamma_0 & \eta \leq 1 \end{cases} \tag{12}$$

γ_0 is the Grüneisen parameter at the initial reference state, p_H is the Hugoniot pressure at the density ρ and is given as

$$p_H = \begin{cases} a_0\mu + b_0\mu^2 + c_0\mu^3 & \mu > 0 \\ a_0\mu & \mu \leq 0 \end{cases} \tag{13}$$

a_0, b_0, c_0 are material parameters fitted from the Hugoniot curves for the uniaxial strain shock wave conditions.

For numerical shock wave computation, the artificial viscosity is needed to stabilize the numerical oscillation [vR50]. The classical artificial viscosity is adapted as the following.

$$q = \begin{cases} \alpha_1 \Delta x^2 \rho \mathscr{D}_H^2 + \alpha_2 \Delta x \rho \mathscr{D}_H & \mathscr{D}_H < 0 \\ 0 & \mathscr{D}_H \geq 0 \end{cases} \tag{14}$$

where α_1 and α_2 are the artificial viscosity parameters, Δx is the element characteristic length. Therefore, the internal energy rate and the Cauchy stress are modified as

$$\dot{e} = \sigma : \mathscr{D} - 3(p + q)\mathscr{D}_H \tag{15}$$

$$\sigma = \sigma_D - (p + q)I \tag{16}$$

3.4.3. Interaction Potential Energy Function

The classical molecular dynamics method was developed for the interaction between discrete particles. It has the following advantages: (1) Accurate physical modeling; (2) Easy implementation; (3) Minimal computational complexity; (4) Symmetry and robustness; (5) Readily interfaces with classical finite elements. The failed material resulting from the continuum damage model is in the form of the debris particles and subjected to the resulting hydrostatic pressure field. The interaction of the failed materials is readily suitable to be modeled by the molecular dynamics method. The interaction potential energy function is appropriately modeled by the soft-sphere pair potential energy function given by the following form,

$$U(r_{ij}) = \alpha(\frac{r_i}{r_{ij}})^{\gamma} \tag{17}$$

where α is the strength of interaction, r_i is the radius of the mass particle ball, and γ is the repulsion parameter. Since the failed material debris particles are subjected to the hydrostatic pressure filed, a soft-sphere pair potential energy function is proposed as the following.

$$U(r_{ij}) = p_j r_j V_i \left(\frac{1}{r_{ij}} \right) \tag{18}$$

where p_j is the hydrostatic pressure of particle j, r_j is the radius of particle j, V_i is the volume of particle i, and r_{ij} is the distance between particle i and j. Therefore, the interaction force between particle i and j is given by the following expression.

$$F_{ij} = -(p_j r_i V_j + p_j r_j V_i) \left(\frac{\mathbf{r}_{ij}}{r_{ij}^3} \right) \tag{19}$$

Most of the existing meshfree methods result in unsymmetric system equations. Therefore, the convergence of the mathematical modeling becomes problem-dependent. Our MD approach results in symmetric system equations with guaranteed convergence.

4. Particle Geometry Reconstruction

Although the above simulation method generates physically-correct results, directly rendering particles as spheres does not convey realistic appearance. Figure 1 shows such an example. There are two major reasons. First, modeling particles as spheres is an acceptable simplification assumption in the simulation process. However, rendering spherical particles makes the phenomena unrealistic. Second, due to the vertex/particle conversion constraints, there will be holes on the boundary surface where the voided elements are not free enough to be released as particles. If the tetrahedral mesh is tessellated fine enough, this may be less of a visual problem. However, an overly-refined mesh is too costly to compute for the simulation.

To provide better visualization, we devise a technique to reconstruct geometric shapes for particles. For each flying-out particle, we find all the voided tetrahedra it is attached to in the original mesh. Tetrahedral splitting is performed on these elements. We first extract one piece from each tetrahedron, then combine these pieces as the geometric shape of that particle. At the same time, pieces are also given back to the element faces connecting the voided elements. For voided elements without particles generated, if the vertices are broken into different pieces, we apply a similar reconstruction strategy. Using this technique, a polygonal boundary surface for the fractured pieces is constructed.

Our reconstruction algorithm is extended from the multi-material interface surface computing algorithm [NF97], where a splitting surface is extracted when the vertices of each element of a tetrahedral mesh are classified as different materials. We take the same spirit, but apply it to a different scenario, where each flying-out or broken vertex of an element is treated as a distinct material. Furthermore, our extension includes timing-varying and topology-changing handling of the data set.

4.1. Tetrahedral Splitting Configuration

We first discuss how to compute the splitting surface in a static volumetric mesh. In many finite-element simulation applications, such as computational fluid dynamics and hydrodynamics, researchers are concerned with reconstructing a boundary surface between multiple materials from simulation results. To represent the material information in the grid cells, each mesh vertex can have an associated tuple $(\alpha_1, \alpha_2, ..., \alpha_m)$ as the material classification, where m is the number of materials. In general, each α_i is a fractional number and it is assumed that $\alpha_1 + \alpha_2 + \cdots + \alpha_m = 1$, as shown in [BDS*03]. In the simplest case, a binary classification is used so that each vertex belongs to exactly one material. The Nielson-Frank algorithm calculates a splitting surface for unstructured grids in such a scenario, which is a generalization of the well-known Marching Cubes (or Marching Tetrahedra) algorithm [LC87]. The edges of a tetrahedron with differently classified endpoints are intersected by the splitting surface. Similarly, the faces of a tetrahedron with three vertices classified differently are assumed to be intersected by the surface in the middle of the triangle. If each of the four vertices has a different class, the boundary surfaces intersect in the interior of the tetrahedron. The resulting mid-edge, mid-face, and mid-tetrahedron intersections are triangulated to form a piecewise approximation surface.

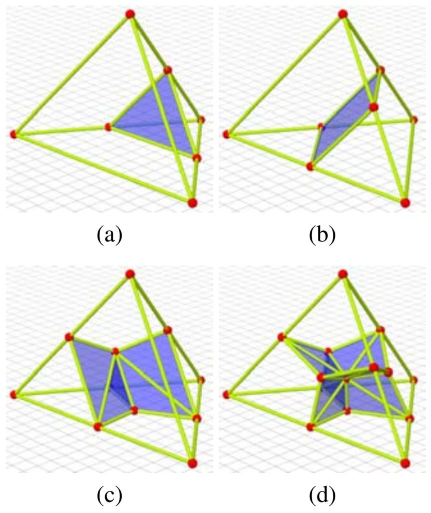
(a) (b) (c) (d)

Figure 4: *The 4 splitting cases of a tetrahedron: (a) 3-1, (b) 2-2, (c) 2-1-1, and (d) 1-1-1-1.*

Since there are only 4 vertices per element, four cases are identified in [NF97]: (1) 3-1, three vertices are of one class and one other vertex is of another class; (2) 2-2, two vertices are of one class and two vertices are of another class; (3) 2-1-1, two vertices of one class and the other two vertices are of second and third classes; and (4) 1-1-1-1, each vertex is of a different class. As the data set is static, once the case number of an element is determined, it will not change. Figure 4 illustrates the four cases.

When we are dealing with time varying data sets, the classification of one element may start from case 1 to case 2, and finally to case 4. If we follow the above splitting configuration, the split objects may exhibit changes in shape, which will cause visual inconsistency in animation. We need coherent splitting so that the split objects can smoothly transit. Therefore, we always use the splitting configuration of case 4 to construct the splitting surface, even though more triangles are used. Figure 5 shows our configuration for 3-1, 2-1-1, and 1-1-1-1 cases from left to right.

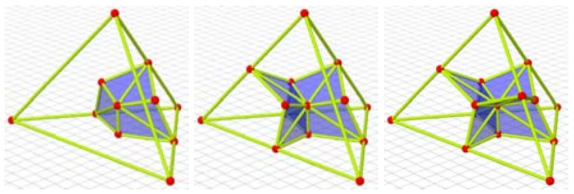

Figure 5: *The three cases for consecutive splitting of a tetrahedron.*

4.2. Dynamic Tetrahedral Splitting

In each time step, we scan the tetrahedral mesh for failed elements. In these elements, we first determine the split case of each element, then determine the split points, and finally we compute the split pieces according to the local coordinate of each component.

Determine the splitting case: The case number of a voided tetrahedron is determined by the sum of the number of particle vertices and the broken vertex sets. Each particle is a disconnected piece. For the remaining vertices that are still in the volume mesh, we further check whether they are directly connected or not. The checking is easily achieved since these vertices are on the boundary surface of an object. By searching the triangle edges of the boundary surface, we can group these vertices into disconnected clusters. For instance, if only one vertex is converted into a particle, it is classified as case 1. If there is one particle vertex and two disconnected clusters of the remaining vertices, it is case 3. Because of the dynamic property of the volumetric mesh, this operation is executed once in each time step.

Determine the split points: Our system keeps a copy of the volumetric mesh in its initial, un-deformed position. Since the element is already voided, in this step we use the original tetrahedron. In a straight-forward way, we can use the mid-edge, mid-face and mid-tetrahedron as the split points, as used in [NF97]. To generate more accurate split positions and let the cutting better conform to the simulation result, we use the scalar damage value computed at each vertex or particle. Then, based on a user-specified damage threshold value, we compute these bisecting positions and store them in the barycentric coordinates of the tetrahedron. The barycentric coordinate values are then used in the next step. We note that for each split point this computation is executed once during the whole process.

Perform local split: As the tetrahedron is already voided, it is incorrect to split the tetrahedron using the current positions of the four vertices. For each splitting component, we need to setup a local, virtual tetrahedron for it. The virtual tetrahedron is built based on the current positions of the vertices in this component. Then, a real split is performed on the virtual tetrahedron. Since we assume a particle is a mass-point and does not have any rotational momentum, the local coordinate is fixed at the time when it is converted from a mesh vertex into a particle. After that, there is no change to its local coordinate. Therefore, we simply record this coordinate for each particle at the time of its creation.

The mesh vertices are translating and rotating during the simulation. To setup the virtual tetrahedron for mesh vertices, we need to find a base triangle in each time step t. In split case 1, the base triangle is formed by the 3 mesh vertices since they are in one cluster. In case 2, for each cluster we need to find a triangle on the boundary surface which shares the two vertices as one of its edges. For each of the single-vertex clusters in case 3 and 4, we need to find a triangle on the boundary surface which shares that vertex. After locating the base triangle on the boundary surface, we compute a differential representation for each vertex in the element that is not used to form the base triangle. In this computation, we use the initial, un-deformed vertex positions at time step 0. The vector $\triangle\,\mathbf{d_0}$ is given by

$$\triangle\,\mathbf{d_0} = \mathbf{d_0} - (\mathbf{a_0} + \mathbf{b_0} + \mathbf{c_0})/3 \tag{20}$$

where $\mathbf{a_0}$, $\mathbf{b_0}$ and $\mathbf{c_0}$ are vertices of the base triangle, $\mathbf{d_0}$ is the vertex to be evaluated, $\triangle\,\mathbf{d_0}$ is the differential vector of $\mathbf{d_0}$. See Figure 6 for illustration. Then, the differential vector $\triangle\,\mathbf{d_0}$ is used to compute $\mathbf{d_t'}$, a virtual, deformed positions of $\mathbf{d_0}$ at time step t:

$$\begin{aligned} \triangle\,\mathbf{d_t} &= k[\triangle\,\mathbf{d_0}\cdot\mathbf{N_x}, \triangle\,\mathbf{d_0}\cdot\mathbf{N_y}, \triangle\,\mathbf{d_0}\cdot\mathbf{N_z}] && (21)\\ \mathbf{d_t'} &= (\mathbf{a_t} + \mathbf{b_t} + \mathbf{c_t})/3 + \triangle\,\mathbf{d_t'} && (22) \end{aligned}$$

where $\mathbf{a_t}$, $\mathbf{b_t}$, and $\mathbf{c_t}$ are the deformed positions, k is the ratio of the perimeter of the deformed triangle $\triangle\mathbf{a_t b_t c_t}$ to the original triangle $\triangle\mathbf{a_0 b_0 c_0}$, and $\mathbf{N_x}$, $\mathbf{N_y}$, and $\mathbf{N_z}$ are the three axis of the triangle $\mathbf{a_t b_t c_t}$'s local frame in world coordinates. Once all the vertices of the virtual tetrahedron are ready, we split the virtual tetrahedron using the barycentric coordinate values computed in advance. The split results are stored as vectors relative to the real mesh vertices. Since there are several tetrahedra sharing one edge, a timestamp is used on each tetrahedron edge to avoid redundant computation and to keep consistency of the split positions.

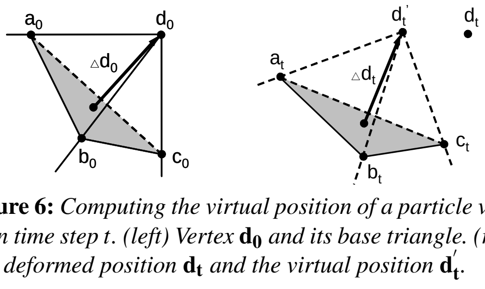

Figure 6: *Computing the virtual position of a particle vertex d_0 in time step t. (left) Vertex $\mathbf{d_0}$ and its base triangle. (right) The deformed position $\mathbf{d_t}$ and the virtual position $\mathbf{d_t'}$.*

5. Experimental Results

We have implemented the simulation and visualization algorithms in C++. The experiments are conducted on a 2.8G Intel Xeon PC with 1GB of RAM. The models used for the simulation are created using NETGEN, a publicly available mesh generation package [Sch97]. We use a linear Complementary Conjugate-Gradient method for contact boundary conditions. In visualization, since the geometry reconstruction algorithm generates many tiny triangles, we perform an online vertex clustering simplification on the output data to reduce the triangle count. In addition, a feature-preserving surface smoothing algorithm is performed on the simplified mesh. The accompanying video contains animations corresponding to the examples. The images are rendered using POV-Ray (`http://www.povray.org`), an open source ray-tracer.

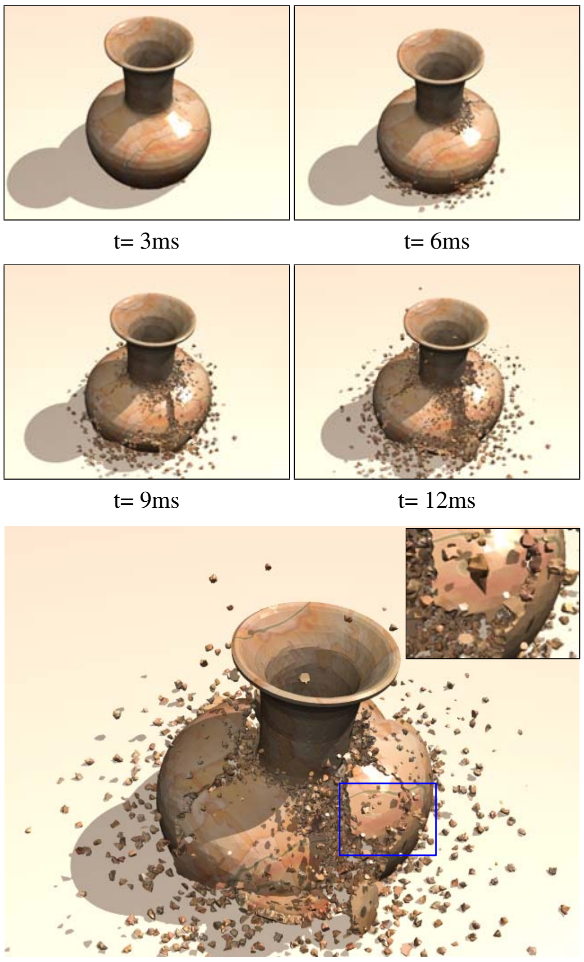

Figure 7: *Simulation of a vase shattered onto the ground, where geometric shapes of the particles are reconstructed and rendered. Images are rendered at different time steps (in simulation time).*

During the computation, we output the results on a fixed time interval of the simulation time. Information about the input tetrahedral element size of each example along with the average time required to compute each time step are listed in Table 1. Instead of tracking the crack tips and maintaining the boundary surface at every step of the simulation, the visualization algorithm operates on the output data only. Assuming that the output time interval is small enough, this does not affect the visualization accuracy and we find that the results are satisfactory. By this means we separate the visualization stage from the simulation stage, therefore making the simulation algorithm simple, clean, and efficient. In fact we have tried most of the meshless methods available in the literature, such as the moving least-square, element free Galerkin, smooth particle hydrodynamics, etc. Due to their un-symmetric nature, each method has its own deficiency such as tensile instability or dramatic time step reduction, etc. The proposed framework overcomes these deficiencies and provides accurate and stable results. Besides, in all examples of this paper, the visualization algorithm takes less than 1 second per frame. We have also counted the extra triangles introduced as the boundary surface for failed elements and particles. On average a particle will introduce around 100 triangles (before simplification), which is determined by the vertex valency in the tetrahedral mesh and the splitting cases.

Figure 7 shows a porcelain vase shattered onto the ground. The vase has a vertical velocity of 25m/s. Since the vase is brittle, it breaks into several large chunks, together with thousands of small pieces. About 25% of the nodes are converted into particles. However, if elements are directly converted, the particle count will be much larger. On the top-right corner of Figure 7, a zoomed-in view of the boxed region in this figure is shown to demonstrate the significant appearance difference with Figure 1 for obtaining the actual geometric shapes of the failed material and debris clouds.

Figure 8 shows a stone plate falling onto the ground with a small angle, where a side-by-side comparison is given. The initial velocity is 20m/s. In the top row, we show the rendering results where spheres are used for particle rendering. In the bottom row, the actual geometric shapes are rendered. As demonstrated by this figure and the companion video, the visual realism is improved significantly.

6. Concluding Remarks

We have introduced a hybrid framework for fracture simulation that generates a large amount of debris particles. We have presented the physics-based governing equations for the hybrid simulation. In visualization, we have further developed a dynamic tetrahedral splitting algorithm to extract geometric shapes for the particles. Our system is physically correct, computationally efficient, and visually realistic.

A general limitation of our current hybrid approach is that a

Table 1: *Simulation performance. The particle numbers are counted at the end of the simulation.*

Model	Tetrahedron #	Node #	Particle #	Frame #	Time step interval (μs)	CPU time per time step (sec.)
Vase	40.4K	13.8K	3445	650	40	37.2
Plate	23.1K	5.4K	644	220	50	21.5

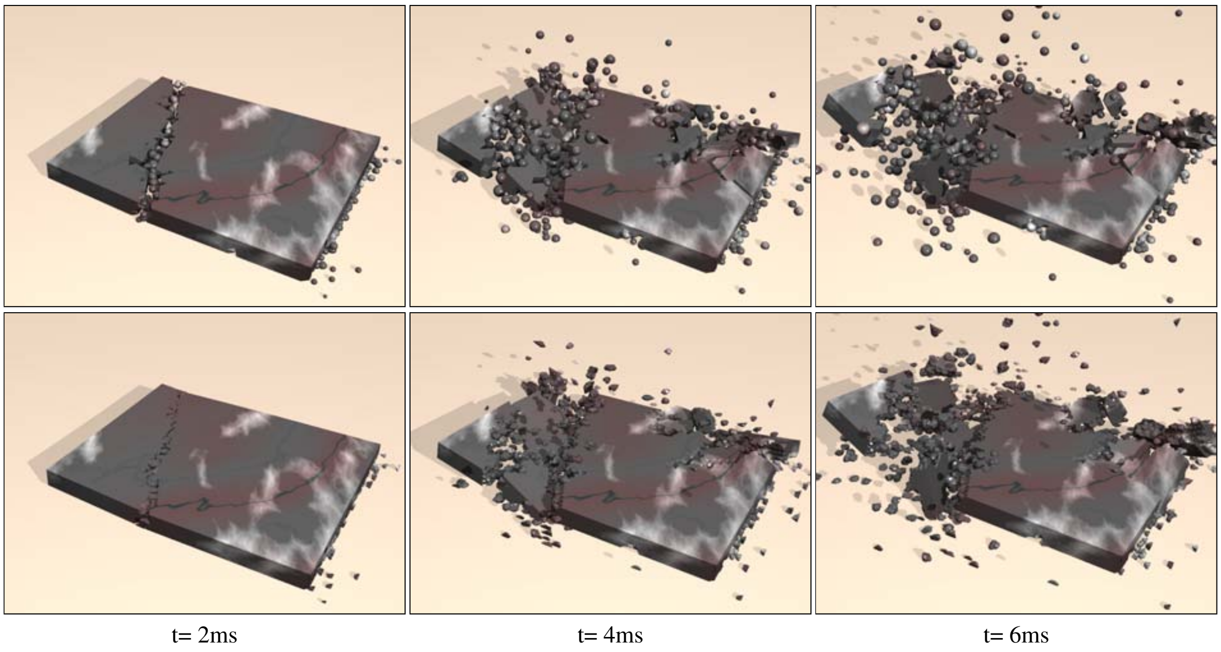

Figure 8: *A stone plate falls onto the ground. The simulation results are rendered using mesh and spherical particles (top row), and geometric shape reconstruction methods (bottom row) at different time steps (in simulation time).*

densely tesselated mesh is required as the input. If the input mesh is too coarse, the simulation results will significantly depart from the correct solution and will look unnatural. Another limitation is that the geometric shapes of the particles are dependent on the initial meshing since the particles are converted from the damaged elements. If one object is tesselated into two different ways, the computed particle shapes are different. This will not be a visual problem as long as the volumetric mesh is adequately refined. Lastly, current implementation of the particles lacks support for angular velocity, which causes an awkward feeling in the demonstration video. We plan to add rotational momentum on particles in our future study.

We envision many practical applications of our system. It could be used for simulating, visualizing, and understanding many phenomena. For example, we could simulate a comet explorer colliding with the comet kernel, which will cause a splendid explosion and generate a large amount of debris particles.

There are a number of avenues for our future work, including:

- Incorporation of adaptive meshing for accelerating simulation.
- Further improvement of the system efficiency using a combination of explicit and implicit time integration.
- Extension of our system to the simulation of other phenomena, such as explosion, fire, smoke, etc.
- Integration of other visualization techniques, such as texture splats, isosurface-based rendering, etc.

7. Acknowledgements

We thank the anonymous reviewers for their comments and suggestions for improving this paper, and Nathan Gossett for proof-reading. The authors are supported by Army High Performance Computing Research Center (AHPCRC) under the auspices of the Department of the Army, Army Research Laboratory (ARL) under Cooperative Agreement number DAAD19-01-2-0014. The content of which does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred. In addition, Baoquan Chen would like to acknowledge the following funding support: NSF ACI-0238486 (CAREER) and NSF DMS-0528492.

References

[AW57] ALDER B. J., WAINRIGHT T. E.: Phase transition for a hard sphere system. *Journal of Chemical Physics 27* (1957), 1208–1209.

[BCHW93] BAMMANN D. J., CHIESA M. L., HORSTEMEYER M. F., WEINGARTEN L. I.: *Structural crashworthiness and failure*. Elsevier, Amsterdam, 1993, ch. Failure in Ductile Materials Using Finite Element Simulations, pp. 1–54.

[BDS*03] BONNELL K. S., DUCHAINEAU M. A., SCHIKORE D. R., HAMANN B., JOY K. I.: Material interface reconstruction. *IEEE Transactions on Visualization and Computer Graphics 9*, 4 (2003), 500–511.

[BYM05] BELL N., YU Y., MUCHA P. J.: Particle-based simulation of granular materials. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (August 2005), pp. 77–86.

[DC95] DESBRUN M., CANI M. P.: Animating soft substances with implicit surfaces. In *SIGGRAPH Proceedings* (1995), pp. 287–290.

[FF01] FOSTER N., FEDKIW R.: Practical animation of liquids. In *SIGGRAPH Proceedings* (August 2001), pp. 23–30.

[FOA03] FELDMAN B. E., O'BRIEN J. F., ARIKAN O.: Animating suspended particle explosions. In *SIGGRAPH Proceedings* (2003), pp. 708–715.

[FSJ01] FEDKIW R., STAM J., JENSEN H. W.: Visual simulation of smoke. In *SIGGRAPH Proceedings* (2001), pp. 15–22.

[HTK98] HIROTA K., TANOUE Y., KANEKO T.: Generation of crack patterns with a physical model. *The Visual Computer 14*, 3 (1998), 126–137.

[JBS00] JOHNSON G. R., BEISSEL S. R., STRYK R. A.: A generalized particle algorithm for high velocity impact computations. *Computational Mechanics 25*, 2/3 (2000), 245–256.

[JBS01] JOHNSON G. R., BEISSEL S. R., STRYK R. A.: An improved generalized generalized particle algorithm that includes boundaries and interfaces. *International Journal for Numerical Methods in Engineering 53* (2001), 875–904.

[LC87] LORENSEN W. E., CLINE H. E.: Marching Cubes: A high resolution 3D surface construction algorithm. In *SIGGRAPH Proceedings* (July 1987), pp. 163–169.

[Liu02] LIU G. R. (Ed.): *Mesh-Free Methods*. CRC Press, Boca Raton, Florida, USA, 2002.

[MBF04] MOLINO N., BAO Z., FEDKIW R.: A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. Graph 23*, 3 (2004), 385–392.

[MKN*04] MÜELLER M., KEISER R., NEALEN A., PAULY M., GROSS M., ALEXA M.: Point based animation of elastic, plastic and melting objects. In *Proceedings of ACM SIGGRAPH Symposium on Computer Animation* (2004), pp. 141–151.

[NF97] NIELSON G. M., FRANKE R.: Computing the separating surface for segmented data. In *IEEE Visualization* (Oct. 1997), pp. 229–233.

[NF99] NEFF M., FIUME E.: A visual model for blast waves and fracture. In *Graphics Interface* (June 1999), pp. 193–202.

[NTB*91] NORTON A., TURK G., BACON B., GERTH J., SWEENEY P.: Animation of fracture by physical modeling. *The Visual Computer 7*, 4 (1991), 210–219.

[OBH02] O'BRIEN J. F., BARGTEIL A. W., HODGINS J. K.: Graphical modeling and animation of ductile fracture. In *SIGGRAPH Proceedings* (August 2002), pp. 291–294.

[OH99] O'BRIEN J. F., HODGINS J. K.: Graphical modeling and animation of brittle fracture. In *SIGGRAPH Proceedings* (August 1999), pp. 287–296.

[PKA*05] PAULY M., KEISER R., ADAMS B., DUTRÉ P., GROSS M., GUIBAS L. J.: Meshless animation of fracturing solids. In *SIGGRAPH Proceedings* (August 2005), pp. 957–964.

[Sch97] SCHÖBERL J.: NETGEN - an advancing front 2D/3D-mesh generator based on abstract rules. *Computing and Visualization in Science 1* (1997), 41–52.

[SWB01] SMITH J., WITKIN A., BARAFF D.: Fast and controllable simulation of the shattering of brittle objects. *Computer Graphics Forum 20*, 2 (2001), 81–91.

[TF88] TERZOPOULOS D., FLEISCHER K.: Modeling inelastic deformation: viscolelasticity, plasticity, fracture. In *SIGGRAPH Proceedings* (1988), pp. 269–278.

[Tru52] TRUESDELL C.: The mechanical foundations of elasticity and fluid dynamics. *Journal of Rational Mechanics and Analysis 1* (1952), 125–300.

[vR50] VON NEUMANN J., RICHTMYER R. D.: A method for the numerical calculation of hydrodynamic shocks. *Journal of Applied Physics 21* (1950), 232–237.

[VXB02] VENTURA G., XU J., BELYTSCHKO T.: A vector level set method and new discontinuity approximations for crack growth by EFG. *International Journal for Numerical Methods in Engineering 54*, 3 (2002), 923–944.

[WC55] WALSH J. M., CHRISTIAN R. H.: Equation of state of metals from shock wave measurements. *Physical Review 97* (1955), 1544–1556.

[ZST03] ZHOU X., SHA D., TAMMA K. K.: An explicit computational formulation for frictional contact/impact problems. In *44th AIAA/ASME/ASCE/AHS Structures, Structural Dynamics, and Materials Conference* (2003), pp. 1772–1782.

[ZST06] ZHOU X., SHA D., TAMMA K. K.: On the new concept and foundations of an arbitrary reference configuration (ARC) theory and fourmulation for computational finite deformation applications - Part II: elasto-plasticity. *Mechanics of Advanced Materials and Structures (In Press)* (2006).