Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2014)
Vladlen Koltun and Eftychios Sifakis (Editors)

# Adaptive Tetrahedral Meshes for Brittle Fracture Simulation

Dan Koschier Sebastian Lipponer Jan Bender

TU Darmstadt, Germany

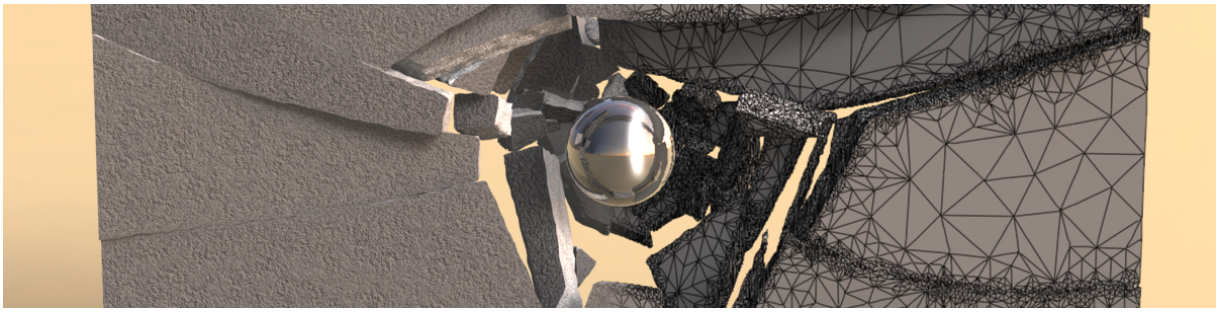

**Figure 1:** *Sphere breaking through a wall using ten refinement steps. The left half shows the final rendered scene. The right half presents the underlying, adaptively refined simulation mesh with high-resolution crack surfaces.*

**Abstract**

*We present a method for the adaptive simulation of brittle fracture of solid objects based on a novel reversible tetrahedral mesh refinement scheme. The refinement scheme preserves the quality of the input mesh to a large extent, it is solely based on topological operations, and does not alter the boundary, i.e. any geometric feature. Our fracture algorithm successively performs a stress analysis and increases the resolution of the input mesh in regions of high tensile stress. This results in an accurate location of crack origins without the need of a general high resolution mesh which would cause high computational costs throughout the whole simulation. A crack is initiated when the maximum tensile stress exceeds the material strength. The introduced algorithm then proceeds by iteratively recomputing the changed stress state and creating further cracks. Our approach can generate multiple cracks from a single impact, but effectively avoids shattering artifacts. Once the tensile stress decreases, the mesh refinement is reversed to increase the performance of the simulation. We demonstrate that our adaptive method is robust, scalable and computes highly realistic fracture results.*

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

## 1. Introduction

Objects made from real world materials fracture when a certain load is applied. While this seems natural and obvious in the context of everyday experience, it is a complex physical phenomenon and an important research topic in computer animation. Real world materials which exhibit only a negligible amount of deformation before they fracture are referred to as brittle materials. Examples from this important class of materials are concrete, glass, stone, and pottery. Many existing techniques for the simulation of brittle fracture are based on a finite element stress analysis using tetrahedral elements and linear shape functions [OH99, MMDJ01, BHTF07]. A fracture occurs when the maximum tensile stress at a vertex or an element center exceeds the material strength. Consequently, the resolution of the initial finite element mesh significantly influences where a fracture can possibly oc-



DOI: 10.2312/sca.20141123

cur and how it propagates. Since the load of an object is generally not known a-priori and therefore also the highly stressed regions in the object, existing methods need a high-resolution finite element discretization in order to provide detailed fracture results. However, a high resolution is only required in regions of high tensile stress where fractures can occur. This motivates our adaptive brittle fracture simulation method which refines the finite element mesh only in such regions using a novel reversible tetrahedral mesh refinement scheme. The usage of an adaptive mesh allows the generation of highly detailed fracture surfaces while the computation time and the memory requirements are significantly reduced. However, high-resolution meshes in combination with existing fracture algorithms suffer from shattering artifacts. In regions experiencing high tensile stress it is very likely that cracks are initiated in multiple neighboring elements. To avoid region-wise shattering we introduce a local non-maximum suppression. Moreover, we propose a stress-rate based method to avoid time-dependent shattering.

Our method treats objects as rigid bodies in the limit of infinite stiffness and performs a fracture analysis only in the case of a collision. This concept was already used in previous works [MMDJ01, BHTF07, GMD12] to increase the overall performance of the simulation. The stress analysis employs a finite element method based on linear shape functions and tetrahedral elements. In case of a collision, our method successively performs a stress analysis and increases the mesh resolution in regions of high tensile stress. The crack propagation algorithm then iteratively inserts a crack at the position of maximum tensile stress and recomputes the changed stress field. The iteration stops when the maximum tensile stress drops below the material strength.

Since an object undergoes only negligible deformation before a crack is initiated, the mesh elements do not degenerate or become skinny. We found it therefore sufficient to split tetrahedra in the initial finite element mesh by the insertion of vertices and to apply a balancing scheme which works by using simple topological operations. In particular, our proposed mesh refinement scheme is very fast due to its conceptual simplicity. We only require to store a small amount of additional data per tetrahedron. From this information the refinement can be reversed and it is guaranteed to obtain the initial mesh again if every refinement operation is rolled back. This works independent from the order of operations and also in the presence of fracture cuts for elements not too close to the mesh boundary. Our method of mesh balancing ensures that the quality of the involved tetrahedra decreases only slightly with each refinement level.

**Our Contributions:**

- An adaptive brittle fracture algorithm for highly detailed fracture surfaces.
- A simple yet effective approach to prevent shattering artifacts using a non-maximum suppression and a stress-rate dependent fracture criterion.
- A novel reversible refinement scheme for tetrahedral meshes which largely maintains the mesh quality and preserves geometric features on the boundary surface.

## 2. Related Work

**Fracture Animation** Terzopoulos et al. [TF88] as well as Norton et al. [NTB*91] modeled fracture simply by breaking the connection between neighboring elements when the forces exceed a certain threshold. Later O'Brien et al. presented a method to fracture brittle [OH99] and ductile [OBH02] materials using a fracture model that relies on a maximum tensile stress criterion, element splitting according to a fracture plane and local remeshing to ensure a conforming mesh. Parker et al. [PO09] introduced a simplified version of this method which is suitable for real-time deformation and fracture in a game environment. Müller et al. [MMDJ01] proposed to treat objects as full rigid bodies in the limit of infinite stiffness for brittle fracture generation and to apply a quasi-static finite element analysis to anchored objects only in the case of collisions. Bao et al. [BHTF07] later improved the plausibility of this approach by the notion of a time averaged stress and a null-space elimination which removes the necessity to anchor objects. Glondu et al. [GMD12] treated objects also as rigid bodies for brittle fracture generation. Their method relies on a precomputed modal analysis, a contact force model for collisions and on a maximum principle stress criterion for fracture initiation. Unfortunately, recursive fracture generation is complicated since the costly, precomputed modal analysis must be recomputed during runtime. Iben et al. [IB06] generate surface crack patterns by evolving a stress field over a triangle discretization over time. Smith et al. [SWB01] proposed to model the shattering of brittle materials by constraints between material points rather than stiff springs. In order to address the degradation of mesh quality due to element splitting, Molino et al. [MBF04] introduced a virtual node algorithm. Later, Sifakis et al. [SDF07] extended the algorithm allowing an arbitrary number of cuts per tetrahedron. To avoid complex remeshing operations Pauly et al. [PKA*05] and Müller et al. [MKN*04] discretize the governing equations of continuum mechanics using a meshless approach and use dynamic resampling during crack propagation for brittle as well as ductile fracture. Wicke et al. [WRK*10] proposed a dynamic remeshing algorithm to address the simulation of purely elastic to highly plastic materials, fracture and large deformations. A hexahedral multigrid approach to simulate cutting in deformable objects was presented by Dick et al. [DGW10]. Their method adaptively refines a coarse simulation grid at the surface of a cutting tool and separates elements along their faces. Several methods to simulate two-dimensional solids adaptively have been proposed in recent publications, e.g. [BD13]. Busaryev et al. [BDW13] simulate the fracture of multi-layered thin plates using adaptive triangle meshes. They propose a stress relaxation method to compute local stress changes after each

cut. This is achieved by solving an elastostatic problem consisting of the elements incident to the cut. However, a local relaxation is not applicable in our case since a single cut changes the stress state throughout the whole object made of brittle material. A level set method for the simulation of ductile fracture was proposed by Hegemann et al. [HJST13] that handles topological changes implicitly through the level set and avoids remeshing. Müller et al. [MCK13] focused on a purely geometry-based approach using approximate convex decompositions for the dynamic destruction of large and complex objects. As the impact's magnitude does not affect the geometry of the shards the physical behavior is less realistic. Later, Schvartzman and Otaduy [SO14] augmented Voronoi diagram based fracture models where the crack generation is guided by the deformation field of the object. A method to simulate realistic fracture of rigid bodies using a novel collision-centered prescoring algorithm was proposed by Su et al. [SSF09]. In contrast to the mentioned methods, our approach is the first approach for tetrahedral meshes that refines regions experiencing great mechanical stresses before cracks are generated.

**Remeshing** Molino et al. [MBTF03] proposed a tetrahedral mesh generation algorithm producing high quality elements from a red-green hierarchy defined on a body-centered cubic lattice, a signed distance function representation of a geometry, and a subsequent compression stage to fit a candidate mesh to the geometry boundary. Later, Labelle and Shewchuck [LS07] presented the isosurface stuffing algorithm, which fills an isosurface with a tetrahedral mesh guaranteeing certain bounds on the dihedral angles. Variants of these approaches have been adopted for the adaptive simulation of fluids, e.g. [WT08] . While these approaches are simple and efficient, they cannot preserve sharp edges or corners, but exactly this characteristic is essential for a realistic simulation of brittle fractures. In order to obtain a conforming adaptive triangulation red-green refinement [BSW83] is often used in practice which is based on a bisection of the six edges of a tetrahedron resulting in four tetrahedra and one octahedron which needs to be further split in either four or eight tetrahedra. A conforming mesh then results from the insertion of green tetrahedra between different refinement levels. In contrast to red-green refinement, our refinement method is more fine-grained, i.e. the number of additional tetrahedra resulting from a refinement operation is smaller, which is desirable since it allows for a better control of the mesh resolution. Among the different existing approaches, the method of Burkhart et al. [BHU10] is most similar to ours. The most striking differences are that our refinement method is reversible and is less complex in terms of the required topological mesh operations. First, our approach does not employ any vertex smoothing rules, since it turned out not to be necessary for the considered input meshes. Hence, it is assumed that the initial mesh is of a sufficient quality such that the quality of the refined mesh is still acceptable for a small deformation finite element stress analysis. Consequently, in order to make any refinements reversible, it is only necessary to keep track of the performed topological operations and vertex insertions. Second, our approach completely avoids the complex edge-removal operations. This is desirable because its inverse operation (multiface-removal) is not unique and would therefore require to store much more information in order to achieve a unique reversion. Furthermore, edge-removal operations performed on any border edges alter the geometry such that geometric features cannot be preserved. In the case of brittle fracture, this is not acceptable as it would result in fracture pieces which do not fit together. Moreover, the exclusive use of simple topological operations makes the mesh adaption more efficient.

## 3. Brittle Fracture Simulation

The phenomenon of brittle fracture arises due to large tensile stresses inside a material. As most brittle materials are deforming very slightly before a crack originates, we decided to model the dynamic behavior using a rigid body simulator. For a general overview on the dynamic simulation of rigid bodies we refer to the survey of Bender et al. [BET14]. High internal stresses are expected to occur in consequence of interactions with other objects. Hence, we perform a stress analysis due to collision events only. This approach is not new and was used before in [MMDJ01, BHTF07, GMD12].

In the following we give a brief overview of the whole fracture generation algorithm. Afterwards a detailed description of every single step is given. To provide a rigid body simulation with robust contact handling we use the open source library Bullet [Bul]. After a collision has been detected, we first estimate the contact duration. Deformations and according internal stresses due to the collision force are determined by the use of a dynamic, linear finite element analysis within a fixed number of time integration steps. After each time step the stresses are inspected for large tensile components. In an iterative procedure regions experiencing large tensile stress are refined according to the scheme presented in Section 4. The stress analysis is repeated until no further refinement is necessary. To avoid excessive shattering we locate local maxima in terms of the largest principle stress while excluding tetrahedral elements with strongly negative stress-rate such that only a single crack originates within a stressed region. As the fracture criterion is based on the Rankine condition [GS11], we assume that a fracture occurs if the tensile stress exceeds a certain threshold $\tau_{crit}$. If the criterion has been met, mesh parts are separated by a fracture surface represented by a signed distance function. Subsequently, we check if stressed regions relax and coarsen their discretization in order to save valuable computation time in the further process. Finally, the mesh is inspected for disjoint parts that will be separated into distinct meshes. In the following we refer to the term "cells" as an alternative to "tetrahedral elements".

**Contact Duration Estimation** In case of a collision between two rigid bodies we first estimate the contact duration according to the Hertz model of sphere-sphere contacts as proposed in [Joh87]. There we replace the sphere radius with the distance between the contact point $\mathbf{p}_{col}$ and the object's center of mass $\mathbf{p}_{com}$:

$$t_d = c\left(\frac{m^2}{E^2\|\mathbf{p}_{com}-\mathbf{p}_{col}\|}\cdot\frac{1}{v_{rel}}\right)^{\frac{1}{5}}, \tag{1}$$

where $c$ is a constant scaling the duration that we usually set to one, $m$ the body's mass, $E$ the Young's modulus and $v_{rel}$ the relative velocity between the contact points in normal direction.

**Finite Element Analysis** After estimating the contact duration we perform a dynamic finite element analysis on the corresponding objects to find regions of stress concentrations. As the rigid body simulator provides impulses $\mathbf{I}$ to resolve collisions, we have to model a force $\mathbf{f}_c$ acting over the time $t_d$ that results in the according collision impulse. We assume that a constant force $\mathbf{f}_c = (1/t_d)\mathbf{I}$ acts on the bodies during the whole collision event. Alternatively, the constant force could be easily replaced by another function as described in [GMD12]. However, we found that the simplified assumption leads to convincing results.

A deformation of an elastic solid can be described by a function $\phi : \Omega \to \mathbb{R}^3$ that maps a point $\mathbf{X} \in \Omega$ from material space $\Omega \subset \mathbb{R}^3$ to a point $\mathbf{x} = \phi(\mathbf{X})$ in world space. In the case of brittle fracture, deformations are small in comparison to the dimension of the object due to the high material stiffness. It is therefore feasible to measure the deformation in terms of linear Cauchy strain $\varepsilon = \frac{1}{2}\left(\nabla\mathbf{u}^T + \nabla\mathbf{u}\right)$, where $\mathbf{u}(\mathbf{X}) = \phi(\mathbf{X}) - \mathbf{X}$ is the displacement of a material point $\mathbf{X}$. For the discretization of $\Omega$, tetrahedral elements with linear shape functions are employed such that $\mathbf{u}(\mathbf{X}) = \mathbf{N}_i(\mathbf{X})\,\mathbf{u}_i$, where the matrix $\mathbf{N}_i(\mathbf{X}) \in \mathbb{R}^{3\times 12}$ interpolates the vertex displacements $\mathbf{u}_i \in \mathbb{R}^{12}$ for the $i$-th element. The constant strain in vector notation over the $i$-th element is given as $\varepsilon_i^V = \mathbf{B}_i\mathbf{u}_i$, where the matrix $\mathbf{B}_i \in \mathbb{R}^{6\times 12}$ is assembled from the partial derivatives of the shape functions $\mathbf{N}_i$. For a linear isotropic material stress and stress-rate over the $i$-th element are given as

$$\sigma_i = \mathbf{C}\mathbf{B}_i\mathbf{u}_i, \qquad \dot{\sigma}_i = \mathbf{C}\mathbf{B}_i\dot{\mathbf{u}}_i, \tag{2}$$

where the matrix $\mathbf{C} \in \mathbb{R}^{6\times 6}$ is the elasticity tensor which is dependent on Young's modulus $E$ and Poisson's ratio $\nu$. According to the principle of virtual work, this yields the element stiffness matrices

$$\mathbf{k}_i = V_i\mathbf{B}_i^T\,\mathbf{C}\mathbf{B}_i, \tag{3}$$

where $V_i$ is the undeformed volume of the $i$-th element. Note that the element stiffness matrices $\mathbf{k}_i$ defined in Equation (3) are not constant if an element undergoes large rotations. To avoid recomputing the $\mathbf{k}_i$ during every collision we precompute them in the original, unrotated configuration. To keep the model consistent we transform the collision forces $\mathbf{f}_c$ using the rigid body rotation $\mathbf{R}$:

$$\mathbf{f}_c^R = \mathbf{R}^T\mathbf{f}_c. \tag{4}$$

The element stiffness matrices $\mathbf{k}_i$ are assembled to the global stiffness matrix $\mathbf{K} \in \mathbb{R}^{3n\times 3n}$ which forms a system of $3n$ ordinary differential equations $\mathbf{K}\mathbf{u} = \mathbf{f}_{\text{ext}} + \mathbf{f}_c^R$ for a tetrahedral mesh with $n$ vertices, where the vector $\mathbf{u} \in \mathbb{R}^{3n}$ consists of vertex displacements from material to world space and where $\mathbf{f}_{\text{ext}}$ denotes the external nodal forces. For a dynamic stress analysis inertia and damping terms emerge such that

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{D}\dot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{f}_{ext} + \mathbf{f}_c^R, \tag{5}$$

where $\mathbf{M} \in \mathbb{R}^{3n\times 3n}$ is a mass matrix and $\mathbf{D} \in \mathbb{R}^{3n\times 3n}$ is a damping matrix. The masses are assumed being lumped in the vertices yielding a diagonal mass matrix $\mathbf{M}$. The damping matrix $\mathbf{D}$ is determined from the damping model of Rayleigh $\mathbf{D} = \alpha\mathbf{M} + \beta\mathbf{K}$ as a linear combination of the mass and stiffness matrix according to the mass and stiffness damping parameters $\alpha$ and $\beta$. We employ the implicit Euler method to integrate the system of ordinary differential equations (5) over the interval $[0,\, t_d]$ in multiple steps. Further, we obtain the solution of the linear system of equations in each time step by a Cholesky decomposition of this symmetric, positive definite matrix and a forward and backward substitution. Finally, the solver matrix remains constant during $[0,\, t_d]$. Hence, the Choleksy decomposition has to be performed just once at the beginning of the FE analysis as long as the mesh is not modified.

**Stress and Stress-Rate Recovery** After evaluating the nodal displacements the stress and stress-rate tensors can be obtained from Equation (2) for each element. Due to the employed linear Lagrangian shape functions, the approximated stress field is constant on each element and exhibits finite jumps between neighboring elements. This issue is a problem in the subsequent stress analysis because it makes the determination of local stress maxima much less robust. We therefore recover a continuous stress field $\sigma^*$ by an $L^2$-projection of $\sigma$ onto the linear Lagrange FE basis which yields a linear system of equations $\mathbf{A}\sigma_{kl}^* = \mathbf{b}$ for each of the six independent stress components $\sigma_{kl}$, where $A_{ij} = \int_\Omega \hat{N}_i\hat{N}_j dv$, $b_i = \int_\Omega \hat{N}_i\sigma_{kl} dv$ and $\hat{N}_i$ are the linear Lagrange nodal basis functions. The linear systems of equations can be avoided by replacing the matrix $\mathbf{A}$ with the diagonally lumped mass matrix $\mathbf{M}$ weighted by the inverse material density $\frac{1}{\rho}$ which leads to

$$\sigma_i^* \approx \frac{\rho}{4m_{ii}}\sum_{c\in\mathcal{I}} V_c\sigma_c, \tag{6}$$

where $\mathcal{I}$ is the set of tetrahedra incident to node $i$. Finally, we perform an eigenvalue analysis on the symmetric, nodal stress tensors $\sigma_i^*$ to determine the greatest tensile stress according to the largest eigenvalue $\tau_i = \max_j \Sigma_{i,jj}$ with $\sigma_i^* = \mathbf{V}_i\Sigma_i\mathbf{V}_i^T$. The stress-rate $\dot{\tau}_i$ is then the corresponding entry

in $\dot{\Sigma}_i = \text{diag}([\mathbf{V}_i^T \dot{\sigma}_i^* \mathbf{V}_i]_{jj})$,where $\dot{\sigma}_i^*$ is obtained from $\dot{\sigma}_i$ by Equation (6).

**Refinement Pass** We assume that the mesh is given in a coarse resolution and therefore apply our refinement scheme locally using a fixed number of refinement steps as described in Section 4. Particularly, we first determine the highest occurring stress $\tau_{\max}$ in regions with stresses exceeding the material strength $\tau_{\text{crit}}$ according to the Rankine condition (see Equation (7)). Subsequently, cells incident to a node $i$ located in the stressed region with stresses $\tau_i > \gamma\tau_{\max}$, $0 \leq \gamma \leq 1$ are refined, where we set $\gamma$ to 0.8 in most cases. Furthermore, Equation (3) has to be evaluated only for the newly arisen cells. Afterwards, we repeat the stress analysis to obtain a more detailed stress field in local regions where stress concentrations are expected.

**Non-Maximum Suppression** When the refinement iterations terminate, we analyze the stress field in order to determine if cracks arise due to large tensile stresses. If this is the case, it is very likely that $\tau_{crit}$ is exceeded by multiple nodes. This would lead to an excessive shattering in the stressed region dependent on the discretization. To avoid this shattering we assume that only a single crack originates in the region's local maximum. Therefore, we use a flood-fill algorithm to determine the region of interest, where we flood only vertices experiencing a tensile principle stress $\tau_i > \gamma\tau_{\max}$ while remaining vertices are treated as border. Afterwards we select the node with the maximum stress to initiate the crack. Then the procedure avoids region-wise shattering while it still allows simultaneous crack initiation caused by different stress concentrations. Finally, the procedure yields a relatively small set of crack originating vertices leading to proper results and to a certain independence from the discretization.

**Fracture Criterion** We use the Rankine condition which states that a brittle material fails if the tensile stress of node $i$ exceeds the material strength

$$\tau_i > \tau_{\text{crit}}. \tag{7}$$

Using the non-maximum suppression according to the previous section we avoided shattering due to highly stressed regions. Furthermore, the stress in the crack's neighboring regions needs a certain time to vanish causing a second type of shattering. Therefore, we extended the criterion in Equation (7) by a second condition

$$\dot{\tau}_i \geq k, \tag{8}$$

where $k \leq 0$ denotes a user-defined constant. Hence, strongly decreasing stresses are not considered for crack initiation.

**Fracture Surface Generation** In our fracture remeshing approach we assume that a fracture surface always originates in a node and that the surface can be represented by an arbitrary geometry with the corresponding eigenvector of the greatest eigenvalue as surface normal at the origin. For the sake of simplicity we assume that the energy causes cracks to propagate until the crack front reaches the object boundary. A possible way to produce a finite propagation is given in [GMD12] using an energy-based fracture stopping criterion.

The fracture surface is represented by an implicit function $s(\mathbf{x}) = 0$, where we presume that $s$ is a signed distance function. Usually a simple plane is considered for $s$, but the fracture geometry can be arbitrarily enriched with any irregularities, e.g. noise functions (see Figure 2). To separate the tetrahedral mesh along the surface we mark all intersected cells in a breadth first fashion and subsequently duplicate these cells to assign them to the according crack flank. The duplication of the cells causes overlapping volumes that we treat as virtual as proposed by Molino et al. [MBF04]. Finally, we store the signed distance in the vertices of the cut cells to reconstruct the crack surface as described in the visualization paragraph. If a cell is intersected more than once, each vertex stores the signed distance with the minimum absolute value.

**Coarsening Pass** Since many tetrahedra can be intersected by the fracture plane, as explained in the previous section, the duplicated tetrahedra can not be coarsened in terms of our refinement scheme. Therefore, we mark these elements as locked. This means that they keep their refinement level. However, all elements that are not locked or affected by the locked ones can be coarsened until they reach their original state. This leads to good transitions from very fine structures near crack surfaces to a coarse resolution in the remaining regions (compare Figure 1).

**Separation of Disjoint Partitions** The mesh is separated into multiple meshes if topologically disjoint regions exist. Each disjoint region is identified using a flood-fill algorithm. The separation is absolutely necessary, since disjoint regions would be mistakenly treated as a single rigid body in the further simulation. Additionally, the separation implicitly decomposes the system of ordinary differential equations (5) into smaller equation systems which improves computation time in further analyses. Finally, we update all quantities that are necessary for the rigid body simulation such as the object's mass, inertia, center of mass etc.

**Visualization** To visualize an unfractured object we simply extract the boundary of the tetrahedral mesh. If cracks arise, the border triangles from fractured tetrahedra are removed and we reconstruct the fracture surface using a signed distance field stored in the vertices of cut cells using a marching tetrahedra algorithm, since the surface is always enclosed by the cut tetrahedra. Note that the signed distance field of multiply cut cells cannot exactly represent the crack surface such that visual artifacts may arise in form of material loss. In order to avoid these artifacts, the approach of Sifakis et al. [SDF07] could be considered.

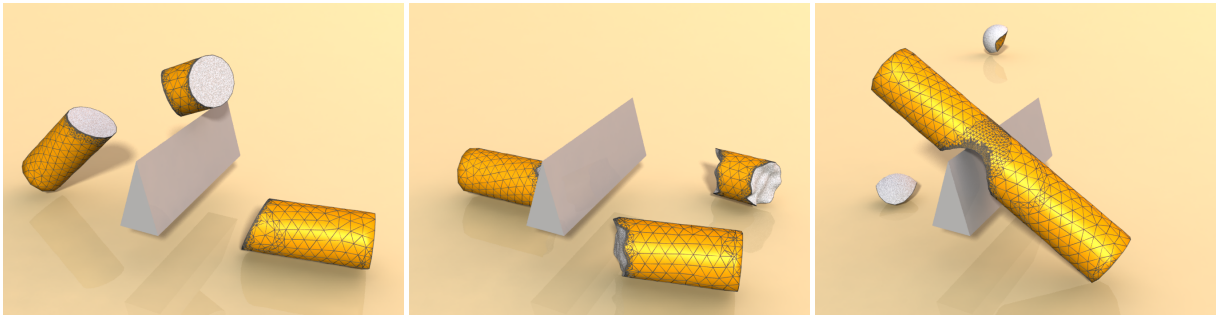

**Figure 2:** *A cylinder is fractured on a sharp obstacle using different fracture surfaces. Left: Planar cracks. Center: Planar cracks enriched using a noise function. Right: Spherical crack geometry.*

## 4. Adaptive Refinement

The basic idea of our novel refinement scheme is that each tetrahedron traverses a maximum of five states within a full subdivision cycle. Note that it is not necessary to perform only full cycles for refinement since all intermediate states provide a valid, manifold topology which gives enhanced control over the granularity of the mesh. For further application the cycle can be applied repeatedly until the desired resolution is reached. The transition between two states is realized by topological operations only such that the scheme can be broken down into a number of split and flip operations as well as their inverse operations. In the implementation we assign a generation index $g_i$ to each tetrahedron $i$ in order to encode the current state and the number of full cycles already applied. Hence, a cell $i$ is in state $(g_i \bmod 5)$ and in cycle $(g_i \operatorname{div} 5)$.

**Refinement Rules** In this section the steps to perform a full refinement cycle are explained in detail. Each of the steps A-E describe the transition between two states (see Figure 3). Some transitions require that neighboring cells are in a specific state. If this requirement is not fulfilled, the neighboring cells are refined until the required state is reached. In order to provide the proposed reversibility of the refinement, the explanation of the steps contains a description of the data that has to be stored per cell. A state transition of one cell generally results in multiple cells of different states. A short description of the transition states is always given in braces according to each step.

**Step A ($\mathbf{0} \leftrightarrow \mathbf{1},\mathbf{1},\mathbf{1},\mathbf{1}$)** To refine a tetrahedron in state 0, we perform a 1-4 split by inserting a vertex at the tetrahedron's geometric centroid yielding four new elements associated with state 1 (see Figure 3A). The four child tetrahedra are obtained by moving one vertex of the original tetrahedron to the centroid and generating three new tetrahedra. It is important to remember which child is the former, original tetrahedron to avoid loosing the refinement information stored in this element. Hence, the child which was created out of the original tetrahedron is tagged after the split. Obviously, the inverse operation can be performed by adapting the corresponding vertex at the tagged tetrahedron and deleting the remaining three tetrahedra.

After finishing step A we have to differentiate between two cases. Each child contains exactly one face of the parent state 0 tetrahedron. If this *inherited face* lies in the interior of the object, we continue with steps B, C and D (see second row of Figure 3). Otherwise, the face lies on the border and we have to perform steps $B^*$, $C^*$ and $D^*$ (see third row of Figure 3).

**Step B ($\mathbf{1},\mathbf{1} \leftrightarrow \mathbf{2},\mathbf{2},\mathbf{2}$)** If the inherited face lies in the interior of the object and if both adjacent tetrahedra are in state 1, the elements are refined by a 2-3-flip. This operation is performed by inserting a new edge connecting the two vertices opposite to the inherited face (see Figure 3B). Then the two original tetrahedra are modified accordingly and a single new element is created.

To ensure that the inverse operation can be applied correctly, the new element has to be tagged in order to remember which cell has to be deleted. Additionally, the edge uniquely shared by the resulting three tetrahedra has to be stored. Note that the inverse operation must guarantee that the untagged cells are not accidentally swapped.

**Step C ($\mathbf{2} \leftrightarrow \mathbf{0},\mathbf{3},\mathbf{3}$)** The edge opposite to the one inserted and stored in step B is trisected regularly (see Figure 3C). The requirement for this operation is that all tetrahedra incident to this edge share the same generation index. After the trisection one element in state 0 and two elements in state 3 are generated. We assign state 0 to one tetrahedron since its quality comes very close to the quality of its original state 0, so no further operations are necessary for this element. The state 0 element is created by moving the vertices of the original tetrahedron and it has to be tagged in order to remember that state 3 and 4 were skipped. For the other two elements new tetrahedra have to be created.

**Step D ($\mathbf{3},\mathbf{3} \leftrightarrow \mathbf{0},\mathbf{4},\mathbf{4}$)** A 2-3-flip is performed to refine two tetrahedra in state 3 sharing a common face (see Figure 3D). Both tetrahedra are required to have the same generation index. The cells are split into three elements by inserting a new edge connecting the vertices opposite

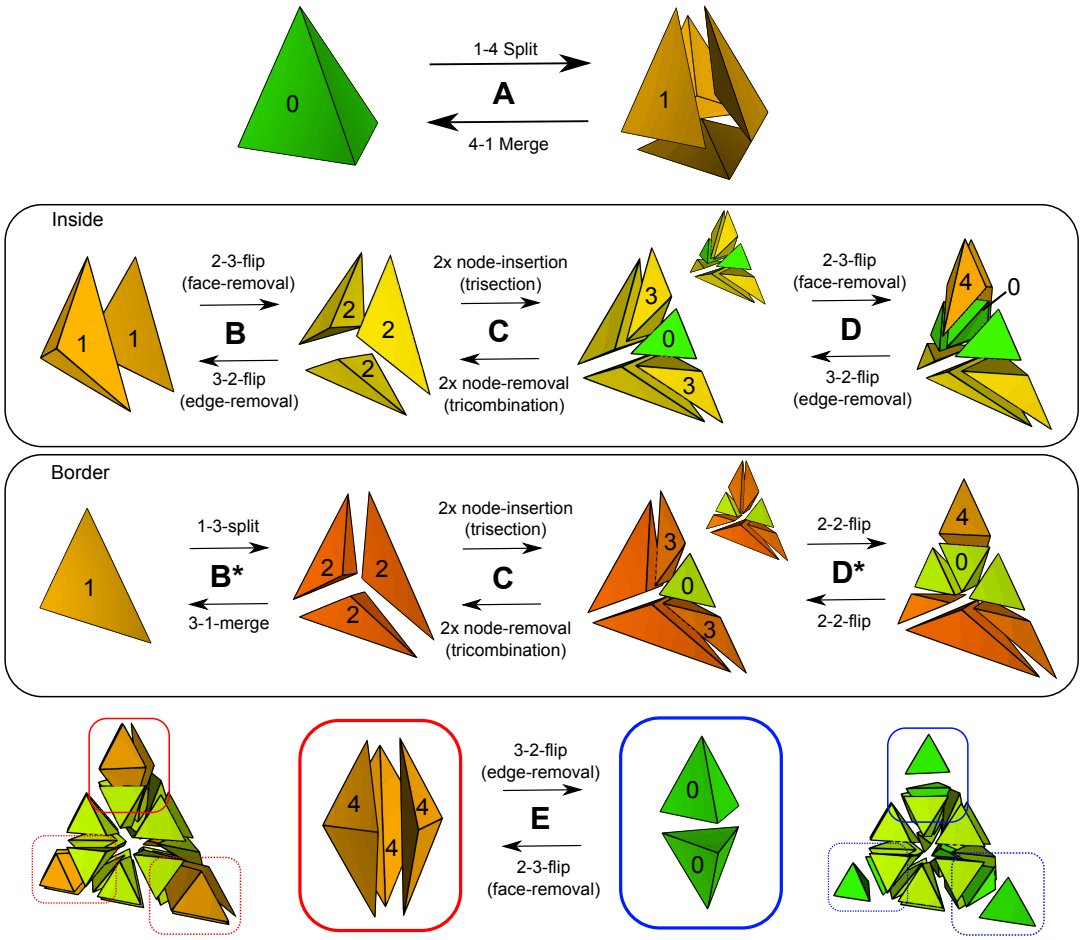


**Figure 3:** *Refinement Pipeline.*

to the common face. The operation results in one state 0 and two state 4 elements. We generate a new tetrahedron for the first element and adapt the original two tetrahedra to obtain the remaining two elements. The first element is tagged and must be deleted in the inverse operation.

If the inherited face of a tetrahedron in state 1 lies on the border, steps B and D have to be adapted:

**Step B$^*$ ($\mathbf{1} \leftrightarrow \mathbf{2},\mathbf{2},\mathbf{2}$)** Since the inherited face lies on the border, there is no adjacent tetrahedron for a 2-3-flip. To perform a similar refinement step, a vertex is inserted at the geometric centroid of the inherited face (see Figure 3B$^*$). This 1-3-split is implemented by adapting the original tetrahedron and creating two new elements. After the split we mark the modified original tetrahedron in order to allow a consistent inverse operation.

**Step D$^*$ ($\mathbf{3},\mathbf{3} \leftrightarrow \mathbf{0},\mathbf{4}$)** In the border case two tetrahedra in state 3 sharing a face are modified by a 2-2-flip (see Figure 3D$^*$). Both elements are required to have generation index 3. Note that in this case two faces of the elements are coplanar. Therefore, no new elements are generated in this step. The 2-2-flip combines the two tetrahedra and splits the geometry into two new ones by a new edge connecting the vertices opposite to the common face. The new elements are in state 0 and 4, respectively, and can be obtained by adapting the original tetrahedra. The inverse operation just requires to reverse the adaption of the elements. No cells have to be tagged.

The last refinement step can be performed for interior and border tetrahedra in the same way:

**Step E ($\mathbf{4},\mathbf{4},\mathbf{4} \leftrightarrow \mathbf{0},\mathbf{0}$)** After applying the refinement steps B$^{(*)}$-D$^{(*)}$ to at least three tetrahedra that result from step A, three state 4 elements are located at each corner of the initial state 0 tetrahedron (see Figure 3E, left). The three elements at a corner can be refined by a 3-2-flip. This flip yields two new tetrahedra in state 0 (see Figure 3E, right). This refinement step is implemented by adapting two of

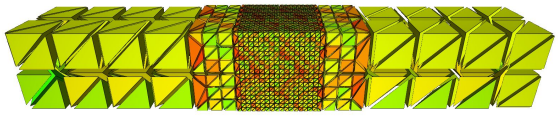

**Figure 4:** *Beam mesh. The center region is refined with either one or two refinement cycles.*

| | Input Mesh | Cycle 1 | Cycle 2 |
|---|---|---|---|
| #Vertices | 106 | 488 | 3862 |
| #Tetrahedra | 223 | 1663 | 17511 |
| Worst min. angle [°] | 30 | 11.36 | 11.36 |
| Avg. min. angle [°] | 43.28 | 41.54 | 41.15 |
| Best min. angle [°] | 70.52 | 70.52 | 70.52 |
| Avg. vertex valence | 8.2 | 10.2 | 11.8 |
| Max. vertex valence | 13 | 22 | 23 |

**Table 1:** *Statistics of initial beam mesh (see Figure 4) and after two refinement cycles.*

the three state 4 tetrahedra and deleting the last one. Note that the deleted cell contains no information required to inverse any operation, since the cell was created in the previous step without storing anything. Hence, it can be safely removed. No cells have to be tagged.

**Information Storage** In each refinement cycle a certain amount of information has to be stored to enable a reversible adaption. First, a counter storing the generation index is needed for each tetrahedron. Second, some refinement steps require information to allow consistent inverse operations:

- Step A: 1 bit to tag the parent cell
- Step $B^{(*)}$: 1 bit to tag newly added cell (or parent cell in case of $B^*$) + 3 bits to store the index of the common edge
- Step C: 1 bit to tag the parent cell
- Step $D^{(*)}$: 1 bit to tag the newly added cell
- Step E: 0 bits.

Summarizing, one can say that each cell requires 7 bits to store the information to roll back the adaption. For example, using a 32 bit integer up to three full refinement cycles (21 bits) in addition to a generation counter (4 bits) can be encoded. Alternatively, a 64 bit integer can encode up to 8 full refinement cycles (56 bits) with a 6 bit generation counter.

## 5. Results and Discussion

**Refinement Statistics** To examine the quality preservation of our refinement scheme we locally refined a beam mesh (see Figure 4) using two subdivision cycles. The according statistics are shown in Table 1. We used the minimum dihedral angle as per tetrahedron quality criterion. The greatest minimum and therefore the best possible minimum angle is $\arctan(2\sqrt{2}) \approx 70.53^\circ$ which is identical to a regular tetrahedron. Obviously, the worst possible angle is 0. Especially, we found that the scheme preserves the average minimum angle very well. Generally, it turned out that the maximum vertex valence is almost constant after one cycle.

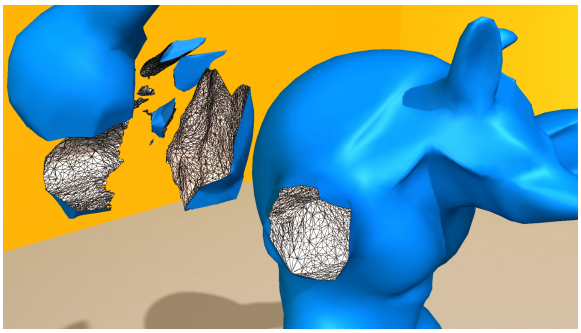

**Figure 5:** *Ball shoots off the limbs of a coarse Stanford Armadillo mesh, providing a high-resolution fracture surface.*

**Figure 6:** *30 tori dropping into a bowl.*

**Scenarios** We applied our method on several scenarios to validate the functionality. We chose examples with different complexities to show the general applicability to many kinds of situations. Note that we chose to perform always five steps of the refinement algorithm at once. This is not mandatory and we decided to use this strategy because if one uses less refining steps the computationally expensive matrix decomposition had to be performed more often, while high-resolution crack surfaces are desired anyway.

In the first scenario the limbs of the Stanford Armadillo are fractured by a projectile. While the mesh is very coarse (approx. 3000 tetrahedra), we achieved a realistic fracture behavior while providing high-resolution crack surfaces using two full refinement cycles (see Figure 5). The second scenario shows a bowl collecting pieces of 30 dropping tori (see Figure 6) while a rigid sphere breaks 10 walls in the third scenario (see Figure 1). Both scenes prove the scalability of our algorithm and that our method works correctly even with many contacts. In a final scene we dropped a series of letters from a certain height (see Figure 7).

| model | total | # frac. | avg. | solver | | | | mesh operations | | | | ref. steps |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | [sec] | | [ms] | init. | fact. | f/b subst. | stress | ref. | coar. | frac. gen. | part. | |
| Armadillo | 22.81 | 91 | 250 | 25% | 24% | 17% | 23% | 2% | 2% | 6% | 1% | 10 |
| tori | 38.27 | 19636 | 1.94 | 20% | 32 % | 12% | 13% | 5% | 3% | 13% | 2% | 5 |
| walls | 76.48 | 18729 | 4.08 | 20% | 20 % | 15% | 17% | 2% | 2% | 22% | 2% | 5 |
| letters | 4.36 | 632 | 6.89 | 21% | 24% | 15% | 15% | 3% | 2% | 16% | 4% | 5 |

**Table 2:** *Timings of example scenarios. For each scenario the total time for the pure fracture computations over the whole domain, the number of fracture algorithm executions and the according average computation time per execution is presented. The fraction of suboperations are denoted in percentage terms. The solver section summarizes initialization and assembling of the solver matrix, its factorization, the front and back substitution for solving and the stress computation. The next section shows the fractions of mesh operations, more precisely the refinement, reverse refinement, fracture generation and separation of disjoint regions. The last column contains the number of refinement steps applied to the objects.*


**Figure 7:** *Scene with letters dropping on a plane.*

All results were carried out on an Intel®Core™i7 CPU 860, 2.8 GHz and 8GB RAM. A summary of computation time performance is given in Table 2. Most of our implementation is not parallelized yet, except the matrix factorization and stress recovery. The most time consuming parts of our method are the solver initialization, factorization and forward/backward substitution consuming about 50%-70% of computation time, as these steps have to be performed after each topological change in the mesh. In comparison to the solver timings, the fraction of the mesh operations are comparatively small.

**Adaptive vs. Uniform High-Resolution** To examine the achieved speed up of the adaptive method we simulated the same scene multiple times with different approaches yielding comparable visual results. We compared a scene using our adaptive approach with reverse refinement to one without the rollback. In both cases we applied two full refinement cycles for adaptivity. Additionally, the results were compared to a scene with an initial high-resolution mesh, but without the usage of any adaptions. The mesh was generated by applying two full refinement cycles as explained in Section 4. Our approach using the refinement without reversing it performed up to 15 times faster in comparison to the static, uniform high-resolution mesh. After additionally switching on the coarsening we gained a speed-up factor of up to 20.

**Discussion** Our adaptive approach is able to produce convincing results at moderate cost. The dynamic behavior of the animated objects is performed using a rigid body simulation which is very fast and stable in comparison to the deformable body simulation in [OH99] and [OBH02] where they depend on very small time steps due to explicit time integration. The stress analysis does not require any anchoring of the collided vertices as in [MMDJ01] or additional computations to remove null spaces as in [BHTF07]. In comparison to the approach of Glondu et al. [GMD12] our fracture algorithm does not depend on a static mesh where a modal analysis is performed on the stiffness matrix and we are able to handle recursive fracture occurrences successively while providing high-resolution crack surfaces independent of the initial mesh resolution.

Using the proposed refinement algorithm the stress analysis was successively enhanced. In comparison to red-green refinement (see e.g. [MBTF03]), we are able to increase the mesh resolution incrementally while the number of tetrahedra is growing slowly. In particular one refinement step leads to 3 to $3m$ new cells, where $m$ is the number of tetrahedra incident to an edge about to be trisected. Furthermore, it is easy to implement since it is based on simple topological operations (avoiding edge-removals) and it is reversible in contrast to the approach of Burkhart et al. [BHU10]. It even preserves sharp features at the boundary geometry which grid-based techniques (e.g. [LS07, MBTF03]) tend to smoothen.

It is most beneficial to use our fracture algorithm with a coarse and equally sized initial mesh in order to gain large speed-ups. However, at the same time this imposes a strong restriction on the amount of surface details a fracture object can have for rendering. In order to serve the different goals of rendering and fracture simulation it is required to use a distinct mesh for each task. In case of a hexahedral simulation mesh Müller et al. [MTG04] proposed a method to fracture embedded surface meshes. We plan to investigate if this approach is applicable to our method. In our fracture generation each crack is propagated until it reaches the border of the object. To achieve a higher realism in future, we will investigate an energy-based stopping criterion for crack propagation as proposed in [GMD12].

## 6. Conclusion

We presented a novel approach for the adaptive simulation of brittle fracture. The algorithm is able to process even coarse meshes while the mesh-dependent stress analysis and fracture surface can be arbitrarily refined using our novel reversible, feature-preserving refinement scheme. We robustly avoid the occurrence of shattering artifacts due to highly stressed regions and are able to speed up further computations with the refinement reversion. Using a dynamic stress analysis we avoid any anchoring of the objects as well as null space eliminations regarding the stiffness matrix. We successively handle recursive fractures and provide the user control over the crack appearance using implicit fracture surface functions. Our results prove that an adaptive refinement significantly improves the generation of highly detailed fractures at moderate costs and the visual appearance demonstrates a great physical plausibility.

**Acknowledgment** The work of the authors is supported by the 'Excellence Initiative' of the German Federal and State Governments and the Graduate School of Computational Engineering at TU Darmstadt.

## References

[BD13] BENDER J., DEUL C.: Adaptive cloth simulation using corotational finite elements. *Computers & Graphics 37*, 7 (2013), 820 – 829.

[BDW13] BUSARYEV O., DEY T. K., WANG H.: Adaptive fracture simulation of multi-layered thin plates. *ACM Trans. Graph. 32*, 4 (2013), 52:1–52:6.

[BET14] BENDER J., ERLEBEN K., TRINKLE J.: Interactive simulation of rigid body dynamics in computer graphics. *Computer Graphics Forum 33*, 1 (2014), 246–270.

[BHTF07] BAO Z., HONG J.-M., TERAN J., FEDKIW R.: Fracturing Rigid Materials. *IEEE TVCG 13*, 2 (2007), 370–378.

[BHU10] BURKHART D., HAMANN B., UMLAUF G.: Adaptive and Feature-Preserving Subdivision for High-Quality Tetrahedral Meshes. *Computer Graphics Forum 29*, 1 (2010), 117–127.

[BSW83] BANK R. E., SHERMAN A. H., WEISER A.: Some Refinement Algorithms and Data Structures for Regular Local Mesh Refinement. *Scientific Computer* (1983), 3–17.

[Bul] Bullet. www.bulletphysics.org.

[DGW10] DICK C., GEORGII J., WESTERMANN R.: A Hexahedral Multigrid Approach for Simulating Cuts in Deformable Objects. *IEEE TVCG 17*, 11 (2010), 1663–1675.

[GMD12] GLONDU L., MARCHAL M., DUMONT G.: Real-Time Simulation of Brittle Fracture using Modal Analysis. *IEEE TVCG 19*, 2 (2012), 201–209.

[GS11] GROSS D., SEELIG T.: *Fracture Mechanics*, 2nd ed. Springer, 2011.

[HJST13] HEGEMANN J., JIANG C., SCHROEDER C., TERAN J. M.: A level set method for ductile fracture. In *Proc. SCA* (2013), ACM, pp. 193–201.

[IB06] IBEN H. N., BRIEN J. F. O.: Generating Surface Crack Patterns. In *Proc. SCA* (2006), pp. 177–185.

[Joh87] JOHNSON K. L.: *Contact Mechanics*, first ed. Cambridge University Press, 1987.

[LS07] LABELLE F., SHEWCHUK J. R.: Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles. *ACM Trans. Graph. 26*, 3 (2007).

[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.

[MBTF03] MOLINO N., BRIDSON R., TERAN J., FEDKIW R.: A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In *Int. Meshing Roundtable* (2003).

[MCK13] MÜLLER M., CHENTANEZ N., KIM T.-Y.: Real time dynamic fracture with volumetric approximate convex decompositions. *ACM Trans. Graph. 32*, 4 (2013), 115:1–115:10.

[MKN*04] MÜLLER M., KEISER R., NEALEN A., PAULY M., GROSS M., ALEXA M.: Point Based Animation of Elastic, Plastic and Melting Objects. In *Proc. SCA* (2004), pp. 141–151.

[MMDJ01] MÜLLER M., MCMILLAN L., DORSEY J., JAGNOW R.: Real-time Simulation of Deformation and Fracture of Stiff Materials. In *Proc. Computer animation and simulation* (2001), Springer-Verlag New York, pp. 113–124.

[MTG04] MÜLLER M., TESCHNER M., GROSS M.: Physically-Based Simulation of Objects Represented by Surface Meshes. *Proc. Computer Graphics International* (2004), 26–33.

[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. *ACM Trans. Graph. 21*, 3 (2002), 291–294.

[OH99] O'BRIEN J. F., HODGINS J. K.: Graphical Modeling and Animation of Brittle Fracture. In *Proc. SIGGRAPH* (1999), ACM, pp. 137–146.

[PKA*05] PAULY M., KEISER R., ADAMS B., DUTRÉ P., GROSS M., GUIBAS L. J.: Meshless animation of fracturing solids. *ACM Trans. Graph. 24*, 3 (2005), 957–964.

[PO09] PARKER E. G., O'BRIEN J. F.: Real-Time Deformation and Fracture in a Game Environment. In *Proc. SCA* (2009), ACM, pp. 165–175.

[SDF07] SIFAKIS E., DER K. G., FEDKIW R.: Arbitrary cutting of deformable tetrahedralized objects. In *Proc. SCA* (2007), Eurographics Association, pp. 73–80.

[SO14] SCHVARTZMAN S. C., OTADUY M. A.: Fracture animation based on high-dimensional voronoi diagrams. In *Proc. Interactive 3D Graphics and Games* (2014), ACM, pp. 15–22.

[SSF09] SU J., SCHROEDER C., FEDKIW R.: Energy Stability and Fracture for Frame Rate Rigid Body Simulations. In *Proc. SCA* (2009), Eurographics Association, pp. 155–164.

[SWB01] SMITH J., WITKIN A., BARAFF D.: Fast and Controllable Simulation of the Shattering of Brittle Objects. *Computer Graphics Forum 20*, 2 (2001), 81–90.

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

[WRK*10] WICKE M., RITCHIE D., KLINGNER B. M., BURKE S., SHEWCHUK J., O'BRIEN J.: Dynamic local remeshing for elastoplastic simulation. *ACM Trans. Graph. 29* (2010).

[WT08] WOJTAN C., TURK G.: Fast viscoelastic behavior with thin features. *ACM Trans. Graph. 27*, 3 (2008), 47:1–47:8.