Fast approximations for boundary element based brittle fracture simulation

David Hahn*
IST Austria

Chris Wojtan†
IST Austria

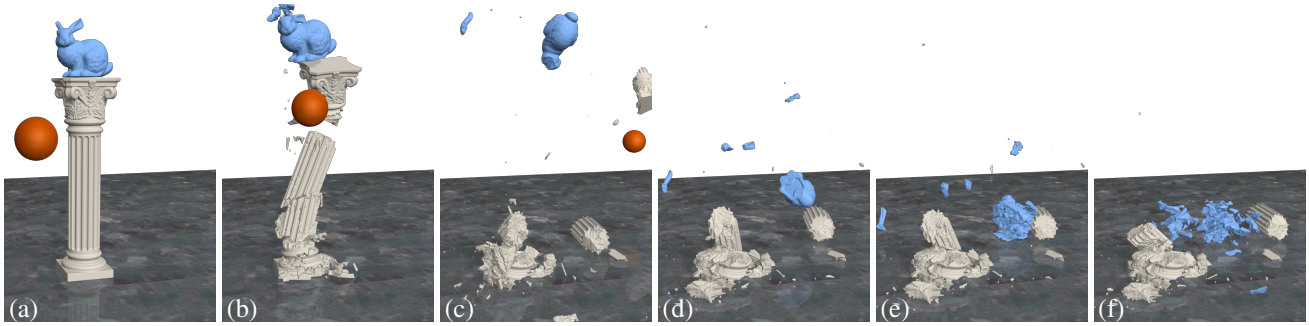

Figure 1: *Shooting a heavy sphere at a column (a): as the column breaks, the topmost fragment is pushed upwards into the bunny, whose ears break off close to the smallest cross-section (b). As the remainder of the bunny falls down it hits a sharp edge of a fragment from the column (d) and fractures again (e). The entire fracture simulation for this scene takes just over 16 minutes on commodity hardware.*

Abstract

We present a boundary element based method for fast simulation of brittle fracture. By introducing simplifying assumptions that allow us to quickly estimate stress intensities and opening displacements during crack propagation, we build a fracture algorithm where the cost of each time step scales linearly with the length of the crack-front.

The transition from a full boundary element method to our faster variant is possible at the beginning of any time step. This allows us to build a hybrid method, which uses the expensive but more accurate BEM while the number of degrees of freedom is low, and uses the fast method once that number exceeds a given threshold as the crack geometry becomes more complicated.

Furthermore, we integrate this fracture simulation with a standard rigid-body solver. Our rigid-body coupling solves a Neumann boundary value problem by carefully separating translational, rotational and deformational components of the collision forces and then applying a Tikhonov regularizer to the resulting linear system. We show that our method produces physically reasonable results in standard test cases and is capable of dealing with complex scenes faster than previous finite- or boundary element approaches.

Keywords: boundary elements, brittle fracture, crack propagation

Concepts: •**Computing methodologies** → *Physical simulation;* •**Mathematics of computing** → *Integral equations;*

*e-mail: david.hahn@ist.ac.at
†e-mail: wojtan@ist.ac.at

SIGGRAPH '16 Technical Paper, July 24-28, 2016, Anaheim, CA,
ISBN: 978-1-4503-4279-7/16/07
DOI: http://dx.doi.org/10.1145/2897824.2925902

1 Introduction

The concept of adding fracture effects to rigid-body simulations has received a lot of attention from the computer graphics community, both in recent research as well as in industrial applications. The most commonly used methods rely either on pre-fractured models, where constraints between individual fragments are removed at the appropriate time, or on pre-defined fracture patterns that are applied when and where collisions occur.

Fully elastodynamic simulations are able to handle both the global motion as well as deformation and fracture. However, they become inefficient for almost rigid materials, as the speed of sound within the material increases, restricting the time step size. Consequently, many previous methods combine elastostatic continuum mechanical models with rigid-body methods to simulate elastic fracture mechanics in a quasi-static sense. These fracture simulations typically use a volumetric discretization such as the finite element method.

Recently, surface integral formulations of linear elastic fracture mechanics have also been investigated. These surface based models promise to reduce the computational cost by reducing the number of degrees of freedom by an order of magnitude over volumetric discretizations at comparable resolution. The resulting dense linear systems are typically harder to solve than their sparse volumetric counterparts though, negating most of the theoretical speed-up even when using fast summation techniques such as the fast multipole method.

In this paper, we present a fracture algorithm based on the boundary element method (BEM), coupled to a rigid-body simulation where the resolutions of the collision mesh, the BEM mesh, and the implicit surface containing the resulting geometry can be freely chosen by the user. The only constraint is that the BEM mesh is coarsest, the implicit surface is finest, and the collision mesh is in between these two. We simulate crack *propagation* at the same resolution as the implicit surface, which allows us to generate detailed fracture surfaces.

We present physics-based approximations that reduce the computational cost of the fracture simulation significantly. In particular, the cost of one time step of our method scales linearly in the *length* of the crack-*front*, while even fast multipole boundary element methods scale linearly in the *surface area*, and finite element methods

scale linearly in the *volume*. We show that this approach reduces the runtime considerably over a standard BEM implementation. Consequently, we can simulate scenes with a huge amount of fractures and detail that would not be practical with any other physics-based method. Finally, our approach can also be combined with an existing BEM solver, where the transition from the full solution to the approximate one is possible at any time, giving rise to a hybrid method. This allows the user to directly control speed vs. accuracy by specifying when to switch from the full solution to the fast approximation.

In summary, our main contributions are:

- Fast approximations of fracture mechanical quantities based on a direct boundary element method, resulting in a linear time simulation of high-resolution crack propagation.
- A hybrid method giving the user direct control over the speed and accuracy of the simulation.
- Coupling these methods to a rigid-body simulation, while carefully treating the resulting Neumann boundary value problem.

2 Related work

The practical approach to simulate brittle fracture of (nominally) rigid objects consists of pre-defining the geometry of all the individual pieces of an object and then "gluing" them back together using constraints. Treating all fragments as rigid bodies allows for fast impulse-based dynamics simulation. When a sufficiently strong collision occurs, the constraints between the fragments are released, giving the impression of fracture; see also [Baker et al. 2011] for details. While this approach gives artists full control over the shape and location of fractures, it also becomes tedious to edit scenes with many fragments of complicated geometry.

Geometry-based methods [Su et al. 2009; Müller et al. 2013] improve upon the standard approach by applying fracture patterns dynamically upon impact. These patterns can be either pre-defined or generated from geometric decompositions of 3d space, such as Voronoi diagrams. While these geometric methods increase the flexibility of the workflow, the realism of the fractures depends entirely on the choice of fracture pattern. Other methods use elastic deformation models to guide geometric fracturing [Iben and O'Brien 2006; Glondu et al. 2012; Schvartzman and Otaduy 2014] or add visual detail to coarse simulation results as a post-process [Chen et al. 2014].

Linear elastic fracture mechanics describes the behavior of breaking objects under the assumptions of continuum mechanics and infinitesimally small deformations. As such, it is a suitable model to build fracture simulations for almost-rigid objects. O'Brien and Hodgins [1999] use a fully dynamic finite element method (FEM) to simulate brittle fractures. However, as the material approaches rigidity, the speed of sound in the material grows, demanding very small time steps or risking artifacts from under-resolved pressure waves. Consequently, many methods such as [Müller et al. 2001; Bao et al. 2007; Glondu et al. 2013] use quasi-static formulations of fracture mechanics to avoid this issue, while objects are treated as rigid bodies when simulating their dynamic motion.

The challenging part of coupling rigid-body dynamics to a (quasi-static) elastic fracture simulation is the transition from the rigid to the deformable model. While elastic models consider *forces*, rigid models typically operate with *impulses* to resolve collisions. These impulses are independent of the time step size (see [Bender et al. 2012] for a review of rigid-body dynamics) as collisions are assumed to be resolved instantaneously in a rigid model. In a deformable model, collisions are resolved gradually within a finite time interval. Converting impulses to forces consequently requires scaling by the collision duration, which cannot be determined from the rigid-body time step. Müller et al. [2001] require the user to specify this duration, while Bao et al. [2007] ignore this issue altogether, referring to the resulting elastostatic stress as "time-averaged", although "time-integrated" would be more accurate as it is not scaled by any characteristic duration. Both Glondu et al. [2013] and Koschier et al. [2014] use a Hertzian contact model to estimate the collision duration, which we also employ (see also Chapter 5 in [Popov 2010] as well as [Chadwick et al. 2012]).

Another interesting aspect of the coupling between rigid bodies and deformable models is how to apply boundary conditions to the deformable model. The main idea is to let the rigid-body simulation handle translations and rotations, while treating deformations separately. Once collision impulses have been converted to forces, the problem that needs to be solved in the deformable model has only Neumann boundary conditions. In this case, the elastostatic solution is only defined up to arbitrary translation and rotation, leading to a rank deficient linear system after discretization. This issue can be avoided by either using a more expensive elastodynamic model as in [Koschier et al. 2014], placing additional Dirichlet boundary conditions [Müller et al. 2001], removing the null-space of the linear system by projection [Zheng and James 2010], or modification of the linear solver [Bao et al. 2007].

The methods described in [Bao et al. 2007; Zheng and James 2010] deal with FEM discretizations and consequently with sparse matrices and iterative solvers. In contrast, our method is based on the direct boundary element method of [Hahn and Wojtan 2015], resulting in a dense linear system which we solve by factorization. Zhu et al. [2015] use an indirect boundary element method, which implicitly removes the null-space by solving for a potential function first and representing displacements as boundary integrals of this potential. Since we use a direct BEM, our treatment of the elastostatic Neumann problem's null-space follows the same principle as [Zheng and James 2010], however, they operate on the nodes of a tetrahedral mesh, while we work on the elements of a triangular surface mesh.

2.1 Brittle fracture simulation

Many of the early works in fracture animation use point-masses connected by either elastic springs or linear constraints; fractures are represented by removing these connections [Terzopoulos and Fleischer 1988; Norton et al. 1991; Hirota et al. 2000; Smith et al. 2001]. Similarly, particle-based discretizations of continuum mechanics include fractures by modifying the interaction forces between particles [Pauly et al. 2005; Stomakhin et al. 2013; Levine et al. 2014].

Fractures are typically represented in FEM simulations by cutting, disconnecting, or duplicating elements [O'Brien and Hodgins 1999; O'Brien et al. 2002; Molino et al. 2004; Parker and O'Brien 2009; Glondu et al. 2013; Koschier et al. 2014], or by additional basis functions in extended FEM [Moës et al. 2002; Gravouil et al. 2002; Kaufmann et al. 2009; Mousavi et al. 2011].

Boundary element methods focus all the computations on the surface of 3d objects [James and Pai 1999; Messner and Schanz 2010; Keeler and Bridson 2014]. Fractures are represented as additional surfaces in the BEM mesh [Aliabadi 1997; Frangi et al. 2002; Hahn and Wojtan 2015; Zhu et al. 2015]. The main advantage of these methods is that they avoid complicated volumetric meshing operations and reduce the required number of degrees of freedom. Hahn and Wojtan [2015] have presented a boundary element based method that can simulate crack propagation at high resolution on

top of a coarse deformation model. They apply rigid-body dynamics only as a post-process, however, and only deal with mixed boundary value problems, while our rigid-body coupling relies on pure Neumann ones. Our implementation is based on their publicly available source code. In Section 3 we extend their method to handle Neumann problems, while details on the rigid-body coupling are given in Section 5. Section 4 describes our approximate method, which considerably improves the theoretical scaling as well as the practical runtime and memory allocation of the fracture algorithm.

3 Boundary element fractures

In this section we summarize the boundary element based brittle fracture simulation of [Hahn and Wojtan 2015] and describe how we extend their method to treat pure Neumann boundary value problems resulting from rigid-body collisions. Given the geometry of an object we first construct a coarse triangle mesh of its surface. We define both the surface displacement and traction fields as piecewise linear and piecewise constant interpolations over this mesh respectively; see [Kielhorn 2009; Hahn and Wojtan 2015] as well as our appendix for details.

This discretization yields the linear system $\mathbf{K}\mathbf{u} = \mathbf{f}$, where $\mathbf{K}$ is effectively a "surface-based" stiffness matrix and $\mathbf{f}$ represents external forces due to boundary conditions. In the case of pure Neumann boundary conditions, this matrix has a null-space with respect to rigid translations and linearized rotations. As $\mathbf{K}$ is constructed from a boundary element formulation, it is also dense.

We first construct a Tikhonov regularizer, which penalizes global translations and rotations, in order to reliably solve this linear elastostatic system. We then proceed to include fracture surfaces in the system. Our regularizer consists of two parts, measuring translation and rotation respectively: $\mathbf{T} := \begin{pmatrix} \mathbf{T}_d^\top & \mathbf{T}_r^\top \end{pmatrix}^\top$, where each part is a $3 \times 3n$ block, where n is the number of surface nodes in the mesh. These blocks are defined such that $\mathbf{T}_d$ measures the average surface displacement $\mathbf{u}_s$, given a piecewise linear displacement field $\mathbf{u}$, and similarly $\mathbf{T}_r$ measures the global surface rotation $\mathbf{r}_s$:

$$\mathbf{u}_s = \int_\Gamma \mathbf{u}(\mathbf{x})\, d\mathbf{x} = \sum\nolimits_e \frac{1}{3} A_e \sum\nolimits_{i=1}^{3} \mathbf{u}_{e,i} = \mathbf{T}_d \mathbf{u}, \qquad (1)$$

and

$$\begin{aligned} \mathbf{r}_s &= \int_\Gamma \mathbf{u}(\mathbf{x}) \times \mathbf{x}\, d\mathbf{x} = \mathbf{T}_r \mathbf{u} \\ &= \sum\nolimits_e \frac{1}{3} A_e \sum\nolimits_{i=1}^{3} \mathbf{u}_{e,i} \times \frac{1}{4}(\mathbf{x}_{e,i} + \mathbf{x}_{e,1} + \mathbf{x}_{e,2} + \mathbf{x}_{e,3}). \end{aligned} \qquad (2)$$

Here $\mathbf{u}_{e,i}$ is the displacement at the i-th node in triangle e, while $\mathbf{x}_{e,i}$ is the material space position of this node, and A_e is the triangle's area. Integration proceeds over the object's surface Γ and we assume (without loss of generality) that the object's center of mass is at the coordinate origin.

We can now use the regularizer $\mathbf{T}$ to turn our rank-deficient linear system into the well-posed problem

$$\mathbf{K}_R\, \mathbf{u} = \mathbf{K}^\top \mathbf{f}, \qquad (3)$$

where the regularized system matrix is $\mathbf{K}_R := \mathbf{K}^\top \mathbf{K} + \gamma \mathbf{T}^\top \mathbf{T}$. The scaling parameter $\gamma := \left(\mathrm{tr}(\mathbf{K})/(3n)\right)^2$ is the squared average diagonal entry in $\mathbf{K}$. Intuitively we want the regularizer to be roughly as "important" as the system matrix. The exact value of γ is not crucial, because the regularizer acts only on the null-space of $\mathbf{K}$ by construction. The boundary conditions, however, must be compatible with a solution free of translation and rotation, or in other words, the right hand side $\mathbf{f}$ must be in the range of $\mathbf{K}$ as pointed out by [Zheng and James 2010]. We provide a similar method to theirs to build such a surface traction field from collision impulses in Section 5.

Now we want to extend Eq. (3) to simulate quasi-static brittle fracture. As in [Hahn and Wojtan 2015] we represent a crack by a single sheet of triangles in the BEM mesh. The unknown quantity on the nodes of this sheet is the crack opening displacement, $\mathbf{\Delta u}$, which is defined as the jump in displacement between the two faces of the crack. Adding these new unknowns to the linear elastostatic system gives:

$$\begin{bmatrix} \mathbf{K}_R & \mathbf{K}^\top \mathbf{C} \\ \mathbf{C}^\top & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{\Delta u} \end{bmatrix} = \begin{bmatrix} \mathbf{K}^\top \mathbf{f} \\ \mathbf{0} \end{bmatrix}, \qquad (4)$$

where the lower part on the right hand side is $\mathbf{0}$ because we assume traction-free fractures. The additional blocks $\mathbf{C}$ and $\mathbf{D}$ describe crack-surface and crack-crack interactions respectively, see Eq. (8) in the appendix for details. Note that the regularizer is not affected by the extension of the system, which means that we do not need to rebuild the regularizer when adding new cracks. When fractures grow, the blocks $\mathbf{\Delta u}$, $\mathbf{C}$, and $\mathbf{D}$ also grow to account for the influence of the new opening degrees of freedom, but $\mathbf{K}_R$ remains unchanged. This means that we can pre-factor the regularized matrix $\mathbf{K}_R$. Solving this system using Schur complements results in $(\mathbf{C}^\top \mathbf{H} - \mathbf{D})\mathbf{\Delta u} = \mathbf{C}^\top \mathbf{b}$ and $\mathbf{u} = \mathbf{b} - \mathbf{H}\mathbf{\Delta u}$, where $\mathbf{b} := {\mathbf{K}_R}^{-1}\mathbf{K}^\top \mathbf{f}$ and $\mathbf{H} := {\mathbf{K}_R}^{-1}\mathbf{K}^\top \mathbf{C}$.

The fracture simulation proceeds in the same way as in [Hahn and Wojtan 2015]: given an unfractured object, we solve for the surface displacement and compute surface stresses per element. Basically, we initiate new cracks orthogonal to the largest principal surface stress, if this stress exceeds the material's strength. However, we use a more accurate surface stress evaluation than [Hahn and Wojtan 2015], which we describe in §4.1. Whenever there are fractures in the object, we solve Eq. (4) for both the surface displacement and the crack opening displacement, $\mathbf{\Delta u}$, and then compute the stress intensities for all crack-front elements using the *displacement correlation technique* [Ingraffea and Wawrzynek 2003]. We propagate the crack-front at high resolution if the stress intensity exceeds the material's toughness. As in [Hahn and Wojtan 2015], we incorporate high-frequency spatial variations of the material's strength and toughness and store the resulting high-resolution geometry as an implicit surface.

We consider this method to be sufficiently accurate within the assumptions of quasi-static linear elastic fracture mechanics. However, it becomes very slow for large numbers of crack-opening degrees of freedom.

4 Fast approximate fractures

Having summarized the boundary element method for quasi-static fracture simulation in the previous section, we now develop a set of approximations that will speed up the crack propagation simulation significantly. In particular the poor scaling of the method described in §3 is due to the cost incurred by the growing matrix block $\mathbf{D}$ in Eq. (4). Our approximations allow us to ignore this block altogether, by sacrificing accurate treatment of crack-crack interactions.

Stress intensity factors (SIFs) describe the singular stress field in the vicinity of a crack-front. The main idea is to estimate these stress intensity factors from the non-singular part of the local stress. This means that we do not need to solve Eq. (4) for the crack-opening displacements. Instead, we can propagate the crack-front using estimated SIFs. Furthermore, as an object fractures, its overall deformation is going to change due to cracks opening; in order to retain

good SIF estimates as the fractures grow, we also account for this additional surface displacement.

Our local stress measure relies only on the object's surface degrees of freedom, excluding fractures. This approximation ignores direct crack-crack interactions (including branching). However, the efficiency gained allows us to handle scenarios with such detailed fracture geometry that these effects will not be visually noticeable in our results.

In the remainder of this section, we first describe how to evaluate local stresses both in the interior as well as on the surface of the object (§4.1), then we present the SIF estimation (§4.2), and finally we discuss how to update the surface displacement due to growing fractures (§4.3).

4.1 Interior and surface stress evaluation

We evaluate stresses in the interior of the domain by integrating over the surface and applying the Kelvin stress kernels as listed in the appendix. However, these kernels diverge as the evaluation point moves to the surface itself, so they cannot be used to evaluate surface stresses. Similarly, the surface stress computation of [Hahn and Wojtan 2015] cannot be used in this case, because it assumes a general 3d stress state. In particular, their stress computation disregards the (known) surface traction and computes a stress purely based on the surface displacements, which would result in a discontinuity between the interior and surface stresses. In this section, we present a surface stress evaluation that accurately includes the known traction and allows for a smooth transition from the interior to the surface stress evaluation. Since the interior evaluation suffers from numerical noise close to the boundary, we transition to the surface stress evaluation at a small distance away from the boundary.

Here we describe how to compute an accurate 3d stress for a surface triangle, given the displacements of its nodes $\mathbf{u}$ (assumed to be linear over the element) as well as the traction $\mathbf{q}$ (assumed to be constant over the element). For a pure Neumann problem, $\mathbf{q}$ is specified as a boundary condition and $\mathbf{u}$ results from Eq. (3). We aim to calculate a surface stress $\boldsymbol{\sigma}$, which is "compatible" with the given traction, i. e. it satisfies $\boldsymbol{\sigma}\mathbf{n}_1 = \mathbf{q}$, where $\mathbf{n}_1$ is the triangle's face normal. Furthermore, we also account for the Poisson contraction in the plane of this triangle caused by the given out-of-plane traction, as illustrated in Fig. 2a. Finally, the surface stress must also be symmetric.

The vectors $\mathbf{n}_1$ (the triangle's surface normal), as well as $\mathbf{n}_2$ (the unit vector along the first edge of the triangle) and $\mathbf{n}_3 := \mathbf{n}_1 \times \mathbf{n}_2$ define a local orthonormal coordinate system. We first compute a 2d stress $\boldsymbol{\sigma}_{2d}$ in the plane spanned by $\mathbf{n}_2$ and $\mathbf{n}_3$. We do this by mapping the nodal displacements into the $\mathbf{n}_2 \times \mathbf{n}_3$-plane and collect them in the vector $\mathbf{u}_{2d}$. The stress computation follows the principle of a basic 2d FEM discretization, i. e. $\boldsymbol{\sigma}_{2d} := \mathbf{S}\mathbf{B}\mathbf{u}_{2d}$, where $\mathbf{S}$ encodes the Hookean stress-strain relation and $\mathbf{B}$ contains the derivatives of the linear shape functions. We then rotate $\boldsymbol{\sigma}_{2d}$ into the global 3d coordinate system to obtain traction vectors $\mathbf{q}_2^*$ and $\mathbf{q}_3^*$. These traction vectors are $(\mathbf{q}_2^*\ \mathbf{q}_3^*) := (\mathbf{n}_2\ \mathbf{n}_3)\ \boldsymbol{\sigma}_{2d}$. In other words, they are defined as the first and second column of the rotated 2d stress respectively.

Given three traction vectors with corresponding orthogonal normals, the 3d stress can be constructed by dyadic summation, as in Eq. (5). However, doing this with $(\mathbf{q}, \mathbf{q}_2^*, \mathbf{q}_3^*)$ might not result in a symmetric matrix, since the tractions built from the in-plane stresses might not be consistent with the given out-of-plane traction $\mathbf{q}$: they need to satisfy $\mathbf{q}_2 \cdot \mathbf{n}_1 = \mathbf{q} \cdot \mathbf{n}_2$ and $\mathbf{q}_3 \cdot \mathbf{n}_1 = \mathbf{q} \cdot \mathbf{n}_3$. Furthermore, the presence of an out-of-plane traction will modify the in-plane tractions due to Poisson's effect. Consequently, we define the corrected in-plane tractions as $\mathbf{q}_i := \mathbf{q}_i^* + \mathbf{n}_1(\mathbf{q} \cdot \mathbf{n}_i) + \mathbf{n}_i(\mathbf{q} \cdot \mathbf{n}_1)\nu/(1-\nu)$, where $i \in \{2, 3\}$, and ν is Poisson's ratio. Finally, we compute the 3d stress by dyadic summation:

$$\boldsymbol{\sigma} := (\mathbf{q}\ \mathbf{q}_2\ \mathbf{q}_3)\,(\mathbf{n}_1\ \mathbf{n}_2\ \mathbf{n}_3)^\top. \tag{5}$$

Verifying that $\boldsymbol{\sigma}$ is symmetric and satisfies $\boldsymbol{\sigma}\mathbf{n}_1 = \mathbf{q}$ is straightforward. In order to derive the factor $\nu/(1-\nu)$ for the Poisson correction, consider a cube under uniaxial tension, σ_z, as illustrated in Fig. 2a: the strain along the z-axis, ε_z, satisfies $\sigma_z = E\varepsilon_z$, where E is Young's modulus. The Poisson contraction along both x- and y-axis is $\varepsilon_x = \varepsilon_y = -\nu\varepsilon_z$. Now consider a triangle on the top face of this cube: the in-plane stress due to Poisson's effect is $\sigma_x = E^*(\varepsilon_x + \nu\varepsilon_y) = -E^*(1+\nu)\nu\varepsilon_z$ with $E^* := E/(1-\nu^2)$ (and analogously for σ_y), which is exactly the result we get for $\boldsymbol{\sigma}_{2d}$. However, since this stress is caused by the out-of-plane tension, σ_z, and there is no external force along either x- or y-axis, we know that the correct result should be $\sigma_x = \sigma_y = 0$. Consequently, we need to add $E^*(1+\nu)\nu\varepsilon_z = \nu E\varepsilon_z(1+\nu)/(1-\nu^2) = \sigma_z\nu/(1-\nu)$ to both σ_x and σ_y.

In summary, we can reliably evaluate the stress on the surface of an object according to Eq. (5), as well as in the interior of the object (using the integral kernels given in the appendix). Because we carefully account for the traction $\mathbf{q}$, our surface stress, $\boldsymbol{\sigma}$, matches the interior stress in an infinitesimal neighborhood. We proceed to estimate local stress intensities based on this stress measure.

4.2 Stress intensity from local stresses

For very simple crack geometries and uniform far-field stress states, it is possible to get closed-form results for stress intensities, as summarized in [Gross and Seelig 2011]. The basic relation between stress intensity factors, K_i, and far-field stresses, σ_i, $(i = 1..3)$ is: $K_i = k\sigma_i\sqrt{\pi a}$, with a constant factor k depending on the crack geometry; according to Table 4.1 (5) in [Gross and Seelig 2011] we chose $k = 1.1215$ for all our examples. Furthermore, σ_i are the projections of the traction vector, $\boldsymbol{\sigma}\mathbf{n}$, onto the crack-front's local coordinates (illustrated in Fig. 2b), where $\mathbf{n}$ is the crack's surface normal. Finally, a is the crack-length; in 3d we measure the area of a crack, A, and assume a planar circular geometry to determine the equivalent length $a = \sqrt{A/\pi}$.

However, these results assume a constant far-field stress, whereas we work with local stress information. When applying the formula above directly we have observed oscillatory crack propagation in our numerical experiments, see Fig. 3. This stems from the fact that using these stress intensity estimates will not result in a crack path that is aligned with an eigenvector of the stress field.

We aim to rectify this behavior by choosing the ratio $r := K_{\mathrm{II}}/K_{\mathrm{I}}$ such that the resulting propagation angle θ aligns the crack's surface

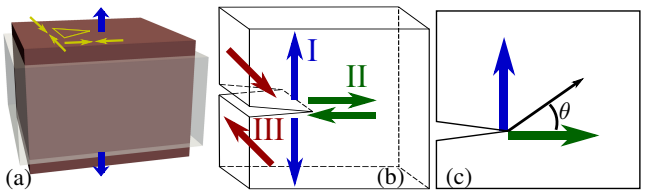

Figure 2: *Illustration of Poisson contraction (a): uniaxial tension (blue arrows) causes the highlighted triangle to contract (yellow arrows). Definition of crack opening modes (b) and the out-of plane crack propagation angle (c).*

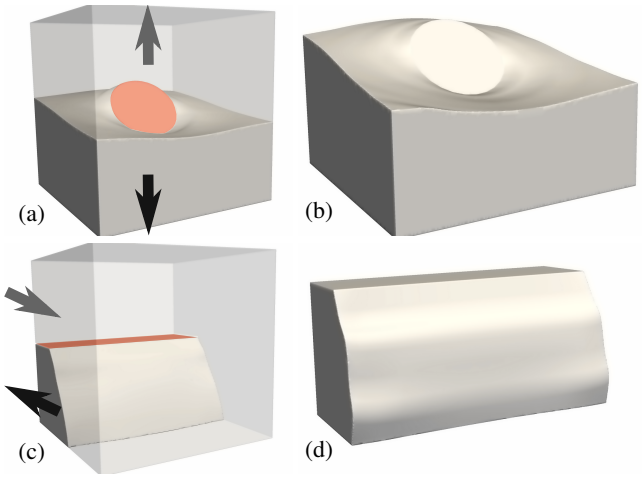

Figure 3: *Comparison of BEM results to the basic SIF estimator for nominally mode-I (a, b) and mode-II (c, d) loaded test cases (arrows) containing pre-defined initial cracks (red). This estimator produces unacceptable oscillation artifacts (b, d) on the crack path.*

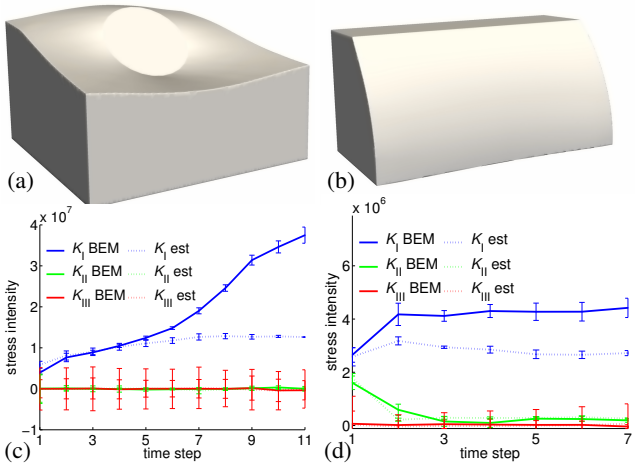

Figure 4: *Aligning the crack path with an eigenvector of the stress field (a, b) removes the oscillations seen in Fig. 3. Stress intensities (c, d) are underestimated as the crack approaches the boundary.*

normal with an eigenvector of the stress. The propagation angle in the crack-front's local coordinate system (see Fig. 2c) is given by [Patricio and Mattheij 2007]

$$\theta = 2\,\mathrm{atan}\left[\left(K_\mathrm{I} - \left({K_\mathrm{I}}^2 + 8{K_\mathrm{II}}^2\right)^{1/2}\right)/(4K_\mathrm{II})\right].$$

Substituting $K_\mathrm{II} = rK_\mathrm{I}$ and solving for the ratio r, which produces a desired angle θ yields: $r(\theta) = \tan(\theta/2)/(2\tan^2(\theta/2) - 1)$, if $\cos(\theta) \geq 1/3$, i. e. θ is less than approx. 70 degrees. We do not explicitly handle larger turning angles, which can only occur under compression. Having computed the desired ratio, we set the stress intensities accordingly and scale them such that $K_\mathrm{I}^2 + K_\mathrm{II}^2 = (\lambda k \sqrt[4]{\pi A})^2$, where λ is the eigenvalue corresponding to the eigenvector we are trying to align with.

Finally, we need to decide how to choose the appropriate eigenvector of the stress. The fracture process should release as much elastic energy as possible; therefore the default choice is the eigenvector corresponding to the largest eigenvalue. However, this eigenvector might not be sufficiently close to the crack normal to satisfy $\cos(\theta) \geq 1/3$. In this case, we choose either the second largest or the most negative eigenvalue, depending on which has the higher magnitude after applying the material's ratio of compressive to tensile toughness to the negative eigenvalue (if such an eigenvalue exists). If two eigenvectors are not sufficiently aligned with the crack normal, we choose the remaining one. Because the stress is symmetric it has orthogonal eigenvectors with real eigenvalues, so at least one of them must be close enough to the crack normal to be a valid choice (as any eigenvector can be multiplied by -1, we have a possible choice every 90 degrees). As shown in Fig. 4, aligning the crack propagation angle with an eigenvector of the local stress removes the oscillation artifacts from the crack surface.

Unfortunately, comparing the estimated stress intensities with the results from the BEM simulation described in Section 3 reveals that the estimates do not grow fast enough as the crack expands, even though the crack's surface area is taken into account; see Fig. 4c, d. We address this issue in the following section.

4.3 Surface displacement update

In the previous section we have described how to estimate stress intensities based on the object's surface displacements and tractions. At the beginning of each fracture simulation, we obtain this information by solving a boundary element system. As the object fractures, however, it effectively weakens and deforms more easily overall. Consequently, in order to get more accurate SIF estimates, it is necessary to update the surface displacements after each crack propagation time step. A full BEM solve, Eq. (4), would give the correct new displacements, but doing so would be too costly. In this section we describe how to approximate the change of the surface displacement, $\mathbf{u}$, due to estimated crack opening displacements, $\mathbf{\Delta u}$.

Given the (estimated) stress intensities, K_i, at the beginning of any one time step, and the new crack-front position at the end of the same time step, we can estimate the crack-opening displacement, $\mathbf{\Delta u}$, at any previous crack-front position by inverting the displacement correlation technique [Ingraffea and Wawrzynek 2003; Hahn and Wojtan 2015]. This process results in $(\mathbf{\Delta u})_i = K_i c\sqrt{2\,d}/(\mu\sqrt{\pi})$, where μ is the shear modulus, d is the distance the crack has propagated during the time step, and the coefficient $c = 1$ if $i = 3$ and $c = (1 - \nu)$ otherwise.

We can now substitute the estimated new opening displacements for the unknown ones in Eq. (4) and move them to the right hand side. Remember that we only aim to update the displacements of the object's surface, so we consider only the first row in Eq. (4). Consequently we find $\mathbf{K}_R\mathbf{u} = \mathbf{K}^\top(\mathbf{f} - \mathbf{C\Delta u})$. We apply an update of the form $-\mathbf{K}^\top\mathbf{C\Delta u}$ to the right-hand side after each time step in order to compute the new surface displacement. Here, the matrix $\mathbf{C}$ describes the influence of new crack-opening displacements on the surface degrees of freedom; as we only need the result of the product $\mathbf{C\Delta u}$, this matrix is never explicitly assembled. We build $\mathbf{C\Delta u}$ only for those crack-front nodes that have propagated during the most recent time step. The (regularized and pre-factored) system matrix $\mathbf{K}_R$ does not change due to these updates.

As shown in Fig. 5, updating the surface displacements in this manner allows the local stress to increase, yielding good stress intensity estimates for basic test cases. Note that this procedure can be applied analogously to mixed boundary value problems such as those discussed in [Hahn and Wojtan 2015] as well.

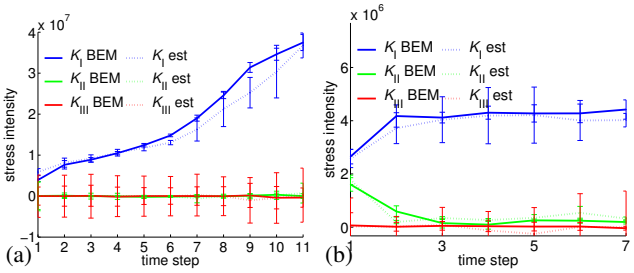

Figure 5: *Comparison of stress intensities obtained by the BEM (solid lines) vs. our estimator (dashed lines): using our right-hand-side updates provides good estimates for both test cases. The corresponding crack surfaces are qualitatively the same as in Fig. 4.*

4.4 Scaling and speed-up

In order to analyze the computational cost of our approximations, let l be the number of nodes on the crack-front: we first need to compute l local stress evaluations. If we evaluate the stress in the interior of the object, this requires an integration over the object's surface, but excluding fracture surfaces. Assuming the number of surface degrees of freedom is n, then evaluating all required stress intensities for any one time step takes $O(ln)$ time. Similarly, the right-hand-side update, $\mathbf{C\Delta u}$, is only assembled for those crack-front nodes that have actually propagated during the time step, therefore it also takes at most $O(ln)$ time. The mesh of the object's surface, however, is *not* changed during the fracture simulation, which means n remains constant and the cost per time step scales linearly with the *length* of the crack-front.

This linear scaling represents a significant improvement over the full BEM solution, where the assembly of the additional matrix block $\mathbf{D}$ in Eq. (4) scales quadratically with the crack area, while the dense linear solve scales cubically. In principle, a fast multipole method [Zhu et al. 2015] could reduce the runtime to (almost) linear in the crack surface *area*. As our approximation removes the need to assemble and store the dense matrix $\mathbf{D}$, this also reduces the required memory considerably.

The practical speed-up depends strongly on the number of fracture elements in the mesh. For simple cases with less than 1000 elements, such as those shown in Fig. 6, the total runtime is dominated by the high-resolution surface tracking and is therefore almost equal for both the full BEM and the approximation. On more complex examples, however, our approximations are considerably faster, as detailed in §6: for example, we obtained a speed-up of approximately 35x for the scene shown in Fig. 8, using about 3x less memory.

Having established that the cost for a single time step scales linearly in the crack-front length, it follows that the computational cost of the entire fracture simulation is linear in the generated fracture *surface area*. Since the cost of producing any explicit surface must at least scale with the surface area, we conclude that our crack propagation method provides optimal scaling.

We can apply our fast estimators described in this section either immediately after solving the unfractured BEM system, Eq. (3), or after computing any number of time steps using the full BEM system, Eq. (4). However, once we do use our estimators, we do not maintain the matrix block $\mathbf{D}$ anymore, so switching back to the full solution would be costly. We construct a hybrid method by allowing the user to set a threshold for the number of elements in the BEM mesh: as long as the number of elements is below this threshold, we use the more accurate BEM solution, and we switch to our fast approximations afterwards.

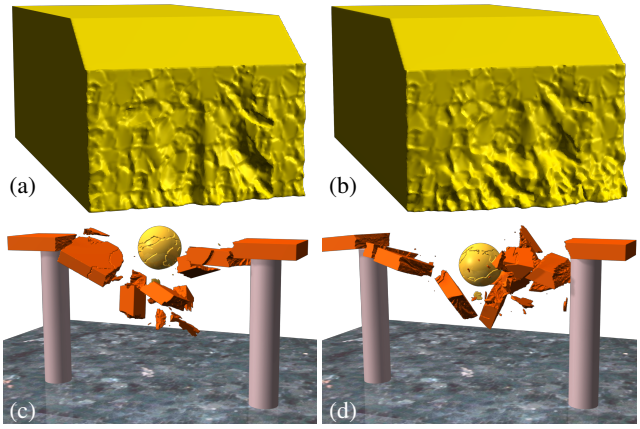

Figure 6: *Comparison of results obtained with the full BEM solution, §3, (a, c) and the fast approximation, §4, (b, d). The fracture locations and surface details are qualitatively equivalent. Please also refer to our accompanying video.*

5 Rigid-body coupling

Having developed our fast fracture method in the previous sections, we now describe how we integrate this method with a standard impulse-based rigid-body simulation. The main parts of this coupling are: computing the right-hand-side vector $\mathbf{f}$ in Eq. (3) and generating the collision shapes for the resulting fragments. Remember that $\mathbf{f}$ must be free of global translations and linearized rotations; otherwise the regularization would counteract these components of the boundary condition and lead to undesirable results. Note that the considerations presented in this section apply equally to both the boundary element fracture simulation (§3) as well as the fast approximate fractures (§4).

5.1 Surface tractions from impulses

In this section we describe how to construct a traction field that is free from global translational and rotational forces and consequently admits an elastostatic equilibrium solution of Eq. (3) and (4). We build this traction field based on the collision impulses after each rigid-body time step. For any one object, we map each of its collision points to the closest element in the BEM mesh, resulting in a piecewise constant input traction field $\mathbf{q}_i$ by applying the Hertzian contact model (we defer the details until the end of this section).

We then solve a quadratic optimization problem with linear constraints to find the smallest correction, $\mathbf{q}_c$, such that the resulting traction field $\mathbf{q} := \mathbf{q}_i - \mathbf{q}_c$ has no global linear force or torque. Similar to Eq. (1) and (2), except that we now use piecewise constant instead of piecewise linear shape functions, we build the linear constraints $\mathbf{f}_s = \int_\Gamma \mathbf{q}(\mathbf{x})\, d\mathbf{x} = \mathbf{A}_f\mathbf{q} = 0$ and $\boldsymbol{\tau}_s = \int_\Gamma \mathbf{q}(\mathbf{x}) \times \mathbf{x}\, d\mathbf{x} = \mathbf{A}_\tau\mathbf{q} = 0$ to enforce vanishing global force and torque respectively.

We can now write our optimization problem as $\min \mathbf{q}_c^\top \mathbf{q}_c$ s. t. $\mathbf{A}_f(\mathbf{q}_i - \mathbf{q}_c) = 0$ and $\mathbf{A}_\tau(\mathbf{q}_i - \mathbf{q}_c) = 0$, or equivalently as the KKT system with Lagrange multipliers $\mathbf{v}$:

$$\begin{bmatrix} \mathbf{I} & \mathbf{A}^\top \\ \mathbf{A} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{q}_c \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix}, \quad \mathbf{A} := \begin{bmatrix} \mathbf{A}_f \\ \mathbf{A}_\tau \end{bmatrix}, \ \mathbf{b} := \begin{bmatrix} \mathbf{A}_f\mathbf{q}_i \\ \mathbf{A}_\tau\mathbf{q}_i \end{bmatrix}. \tag{6}$$

The Schur complement solution of this system is: $\mathbf{q}_c = -\mathbf{A}^\top \mathbf{v}$ and $\mathbf{v} = -(\mathbf{A}\mathbf{A}^\top)^{-1}\mathbf{b}$. Note that the matrix $\mathbf{A}\mathbf{A}^\top$ has dimension 6×6 regardless of the number of triangles in the mesh. This matrix can be built directly, requiring only one pass through the element list. The projection used by [Zheng and James 2010] applies a similar idea to a tetrahedral mesh. Having found the translation- and rotation-free piecewise-constant surface traction field, $\mathbf{q}$, closest to the input tractions, $\mathbf{q}_i$, we assemble the right hand side, $\mathbf{f}$, of Eq. (3) according to Eq. (9) in the appendix.

Each collision contributes the traction (or force per area) $J\hat{\mathbf{n}}/(t_c A_e)$ to the entry corresponding to element e in the input traction vector $\mathbf{q}_i$. Here A_e is the area of the element, J is the impulse transferred by the collision, $\hat{\mathbf{n}}$ is a unit vector pointing along the direction of approach (in the object's local coordinate system), and finally t_c is the contact duration.

The collision impulse is independent from the rigid-body time step, therefore we need to estimate the contact duration, t_c, for each collision. Hertz [1881] describes the collision of two elastic spheres, or equivalently of one elastic sphere with a rigid plane. This sphere is described by an effective elastic modulus $E = [(1-\nu_1^2)/E_1 + (1-\nu_2^2)/E_2]^{-1}$, radius $R = (R_1^{-1} + R_2^{-1})^{-1}$, and mass $m = (m_1^{-1} + m_2^{-1})^{-1}$. Each of these three effective quantities combines the parameters of both objects involved in the collision. The collision duration is then $t_c = 2.87[m^2/(E^2 R v)]^{1/5}$, where v is the velocity at which the two objects locally approach one another. We refer the interested reader to Chapter 5 in [Popov 2010] for a detailed derivation.

We obtain the required parameters as follows: elasticity parameters and densities are user inputs and constant per object, the local velocities of the contact points, as well as the transferred impulse are taken from the rigid-body simulation, and the effective radii are the inverse of the mean curvature at the contact locations. The volume of each object, required to compute its mass, is measured on the high-resolution implicit surface.

5.2 Fragment generation

Once the fracture simulation for any one rigid body has been completed, we need to identify new fragments and update the rigid-body scene accordingly. Similar to [Hahn and Wojtan 2015] we use a level set to represent the high-resolution surface of each object, including all fractures, and use a breath-first-search algorithm to find connected components, i. e. fragments.

We call the rigid body which has just fractured the "parent" object. At the beginning of the rigid-body simulation the total volume of each object is computed, and this information is copied from parent to fragments. We refer to this as the "original volume" and classify fragments into the following four groups based on their relative volume:

(a) very small fragments (0.5% or less of the original volume): for efficiency reasons these are not allowed to fracture any further in our implementation and are represented by their convex hull in the rigid-body scene;

(b) small fragments (0.5–2% of the original volume): cannot fracture either, but use a (possibly non-convex) mesh for collision detection;

(c) large fragments (more than 2% of the original, but less than 95% of the parent's volume): can fracture again and have both a mesh for collision detection as well as a (coarser) mesh on which the BEM computations of future collisions are handled;

(d) very large fragments: if we find a fragment that has more than 95% of its immediate parent's volume, we do not start a new BEM instance and instead retain the one from the parent object, while the rigid-body collision shape is updated. This way we avoid the overhead incurred from creating a new BEM mesh as well as building, regularizing and pre-factoring the matrix $\mathbf{K}_R$. If there is no such fragment, the parent object is deleted along with all its associated BEM data.

We mostly classify fragments based on the object's original volume rather than the volume of their immediate parent object because it retains a better notion of scale: a fragment of a particular size should be treated in the same way regardless of whether it has broken off the original object right away, or after a series of intermediate fracture events.

Except for large ones, all other fragments are easily handled by adding (or updating) their shape in the rigid-body scene and copying all material parameters from the parent. Positions and velocities of all fragments are set such that they match the motion of the parent object *before* the collision happened. Once all fragments are in place, we repeat the rigid-body time step that caused the collision (but without starting any new fracture simulations), allowing the newly created fragments to respond to the collision.

5.3 Fragments containing incomplete cracks

For large fragments, we need to construct a boundary element mesh and assemble the linear system as described in §3. As each fragment may contain cracks that do not cut it into separate pieces (yet), we also have to represent these cracks in the BEM mesh to allow them to propagate in future fracture simulations; we refer to them as "incomplete" fractures.

We first obtain a BEM mesh by extracting a high-resolution triangle mesh from the fragment's level-set representation, excluding all incomplete fractures, and coarsening according to a user specified resolution. Then we identify all incomplete fractures and copy their level-set representation, as well as all elements from the parent's BEM mesh that are inside of the fragment.

This way the representation of an incomplete fracture in the fragment's BEM mesh is a subset of its representation in the parent's BEM mesh; we take care to ensure that this subset is a collection of manifolds with boundary, i. e. the new crack-front, which is the bounding curve of this subset of triangles, has exactly two edges meeting at each node. Wherever the new crack-front coincides with the old one, we copy high-resolution crack-front data from the parent as well. This treatment allows "old" fractures to propagate further due to subsequent collisions of the fragment.

6 Results

In this section, we first discuss test cases demonstrating that our estimated stress intensity factors produce results that are acceptably close to the BEM solution. We also show that our estimator is much faster than solving the full BEM system, which enables us to simulate larger scenes with many more fragments in a reasonably short time. Finally, we compare the distribution of fragment volumes generated by our simulation to theoretical results, before introducing more artistic examples that are also featured in our accompanying video.

In Section 4 we introduced the main components of our SIF estimator, namely alignment with an eigenvector of the stress field, and the update to the right-hand side of the linear elastostatic system on the object's surface. Figures 3 and 4 show that these components are necessary to avoid both oscillation artifacts on the result-

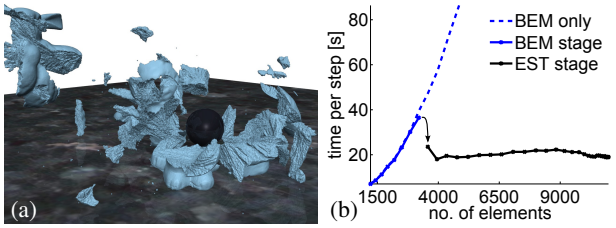

Figure 7: *Smashing a bunny (a) using our hybrid method. The CPU time required to compute each time step (b) increases dramatically when using the full BEM solution, but immediately becomes independent from the number of elements in the mesh once we switch to our fast estimators. The "BEM only" control run is plotted until it reaches a memory consumption of 4GB.*

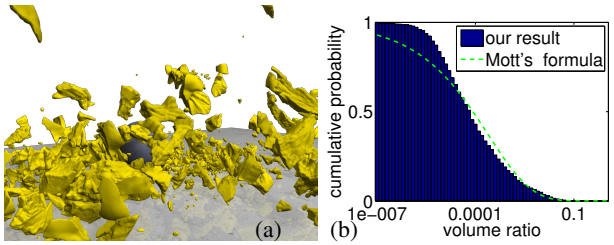

Figure 8: *Smashing a weak bunny using our SIF estimation method. The distribution of fragment volumes is in reasonable agreement with theoretical predictions.*

ing crack surface and underestimation of stress intensities as the crack-front approaches the surface of the object. Figure 5 shows that with our modifications, we are able to obtain good SIF estimates for basic test cases. The mode-I test uses mixed boundary conditions, while the mode-II case uses only Neumann boundary conditions. In our experiments mixed boundary conditions on the mode-II case resulted in underestimation of K_{I}, making the material appear tougher. We accept this limitation, as we use pure Neumann coupling in all our rigid-body scenes.

Figure 6 visually compares the full BEM solution and our fast approximations for two low-complexity, yet non-trivial, examples: the 3-point-bending test of a notched bar, as in [Hahn and Wojtan 2015], and a similar setup for a bar without notching. In both cases our approximations yield qualitatively equivalent results, providing further evidence that our estimates are physically reasonable. The notched bar example, however, shows that the compressive zone at the bottom is not quite as well resolved by the fast method, resulting in a fracture surface that is not as perpendicular to the length-axis of the bar.

In general, simulating very few clean cuts through an object requires a more accurate deformation solution, which means that the full BEM, or our hybrid method are better choices than the fast estimators. The approximation errors become visually unnoticeable as the number of fractures increases.

Figure 7 demonstrates the efficiency of our SIF estimation method compared to a BEM approach: we use a setup similar to the example in Table 1 (6) of [Hahn and Wojtan 2015], smashing the bunny by applying mixed boundary conditions and one-way rigid-body coupling. The example shown here uses a weaker material and higher boundary traction to produce more fractures than theirs. We use our hybrid method, solving the full boundary element system as long as there are less than 3000 elements in the mesh, and applying our fast SIF estimation method afterwards. The CPU time required to compute each crack propagation time step is shown in Fig. 7b. While the BEM quickly becomes very slow, our estimator is independent from the total number of elements in the mesh: it only scales with the length of the crack-front, which actually reduces slightly as the crack-front reaches the object's surface in some places earlier than others. The simulation terminates when all cracks have reached the surface.

By applying our two-way rigid-body coupling and SIF estimator, and choosing an even weaker material for the bunny, we can extend this example to produce over 1000 fragments in 4 rigid-body time steps. We show the distribution of fragment volumes in Fig. 8 and compare it to Mott's formula, a widely accepted theoretical model for fragment size distribution, which is given by $P(V) = e^{-\sqrt[3]{\zeta V}}$, $\zeta = 6/\bar{V}$. Here $\bar{V}$ is the average fragment volume (see also Eq. (20) in [Elek and Jaramaz 2008]). Figure 8b plots the probability that a randomly selected fragment has a relative volume equal or larger than a given ratio. Fragment volumes are measured relative to the original object's volume. The volume distribution of our result is in good agreement with theoretical predictions overall, except that our result contains less very small fragments than predicted. We believe this deviation to be mostly due to the resolution limit of the implicit surface on which we store the high-resolution geometry; i. e. fragments much smaller than the voxel size cannot be represented.

As described in §4.4, the speed-up (i. e. the reduction of runtime spent on fracture simulation for a particular scene) depends strongly on the number of elements used to represent fractures. Our basic comparisons in Fig. 4 and 6 take about the same time for each method due to the very low element count. For the "Bowl" example in our video, a scene of medium complexity, the fast approximate fracture simulation runs about 10x faster than the full BEM solution. Similarly, our hybrid method applied to the scene in Fig. 7 reduces the runtime roughly by a factor of 14, i. e. the simulation completes in less than 15 minutes instead of almost 3.5 hours required for the full BEM. On our largest example in terms of fracture elements, Fig. 8, the fast approximation is about 35x faster than the BEM (about 22 minutes instead of 13 hours). All of these timings refer only to the fracture simulation, excluding time spent on rigid-body dynamics.

More artistic examples are shown in Fig. 1 and 9, as well as in our accompanying video. Please refer to Table 1 for an overview of our method's performance on these scenes. In many of these examples, fracture simulation accounts for about 70% of the total CPU time, while rigid-body time stepping accounts for the other 30%.

7 Limitations and future work

In our current implementation we use a basic single-threaded rigid-body simulation based on Bullet [Coumans 2015] and our coupling to the fracture simulation considers collisions only. It might be interesting to also include inertial forces, such that fast spinning can cause fracture. Bullet's collision detection is prone to popping artifacts when performed on meshes. This issue could be improved, and also sped up, by convex decomposition [Mamou and Ghorbel 2009]. For example, the scene shown in Fig. 1 contains geometrically interlocking fragments, which require higher resolution collision shapes and consequently slow down the collision detection step of the rigid-body solver considerably. Because we did not implement known rigid-body optimizations, the time spent on rigid-body dynamics (marked with '*' in Table 1) should not be considered to be representative in this case. We manually removed rigid-body

Table 1: *Performance overview: timings were obtained on a 4x3.2GHz CPU with 4GB available memory. Columns: method of fracture (when using the hybrid method the number of elements beyond which we switch to the fast estimators is given in brackets), number of fragments produced f, total time spent on fracture simulations t_f, number of rigid-body collisions causing new fractures r, maximum number of triangles contained in a fracture simulation due to a single collision m_{max}, total number of triangles involved in all fracture simulations m_{all}, time spent on rigid-body dynamics t_{rb}, number of rigid-body time steps, rigid-body frame rate per seconds $1/dt$. Scenes are listed in order of appearance in our accompanying video. We do not include rigid-body timings for the example in Fig. 7, because the dynamics were simulated separately after the fragments had been generated.*

Scene	Method	f	t_f	r	m_{max}	m_{all}	t_{rb}	Steps	$1/dt$
Breaking Π (Fig. 6)	BEM	122	388.58s	17	1004	9645	79.97s	1000	250
Breaking Π	EST	100	292.73s	13	933	7552	140.60s	1000	250
Breaking Π	HYB (400)	168	461.36s	16	1225	9705	108.76s	1000	250
Bunny one-way (Fig. 7)	HYB (3000)	185	892.25s	1	10981	10981	(post-process)		
Brick breaking	HYB (300)	60	216.11s	26	412	7525	78.18s	500	2000
Bunny two-way (Fig. 8)	EST	1099	1328.17s	5	15381	35555	565.92s	500	250
Column (Fig.1)	EST	367	966.81s	18	7980	31529	*2628.18s	1000	250
Armadillos (Fig. 9)	EST	148	885.95s	14	1792	18079	1710.42s	600	250
Window (Fig. 9)	EST	455	1256.85s	7	6459	31186	1731.38s	1000	250
Bowl	EST	139	845.99s	14	3250	19504	351.32s	500	250
Falling boxes	EST	654	706.43s	77	704	26879	1498.49s	1000	500

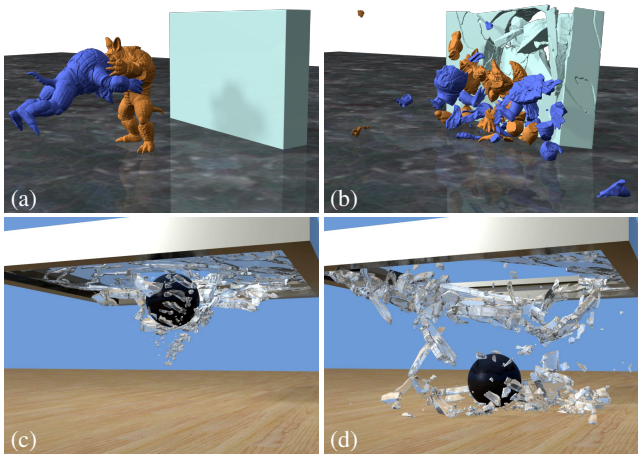

Figure 9: *Arguing armadillos cause collateral damage (a, b). Even though boundary element based methods are most efficient for objects with large volume to surface ratios, our method is capable of fracturing a thin (but volumetric) window (c, d).*

popping artifacts from some of our examples by post-processing the motion, because collision resolution problems are orthogonal to our work and we did not want them to interfere with the presentation of our main research contribution.

We have shown that the SIF estimation works well for crack propagation, but branching is not handled at the moment. This could be done by adding an explicit branching criterion, for example based on the remaining two eigenvalues of the local stress, which correspond to eigenvectors close to the crack surface's tangent plane. We also ignore direct crack-crack interactions in the estimator, although some analytical results exist describing these effects (sometimes referred to as "transmission factors"). It might be possible to further speed up the initial BEM solution by using a pre-computed proxy shape, similar to the "sound proxy" of [Zheng and James 2010]. It could also be interesting to include their "fracture impulses" in the rigid-body dynamics after generating fragments.

In summary, we have presented a physics-based simulation method for brittle fractures using fast estimation of stress intensities to speed up crack propagation, which is simulated at high resolution. Both deformations and collisions are handled on coarser meshes. All three resolution levels can be chosen by the user. This results in a practical and efficient physics-based simulation of rigid-body fracture, which is both faster in practice and scales better than previous finite- or boundary element methods. In light of the fact that any method that generates a fracture surface mesh must iterate over each node in that mesh at least once, we consider the scaling of our crack propagation algorithm to be optimal.

Acknowledgments

We cordially thank everyone who contributed to this paper, especially Prof. M. Schanz and his group for providing and supporting the HyENA library, the OpenVDB online community, as well as all proof-readers, in particular Stefan Jeschke. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 638176).

References

ALIABADI, M. 1997. Boundary element formulations in fracture mechanics. *Applied Mechanics Reviews 50*, 2, 83–96.

BAKER, M., ZAFAR, N. B., CARLSON, M., COUMANS, E., CRISWELL, B., HARADA, T., AND KNIGHT, P. 2011. Destruction and dynamics for film and game production. In *ACM SIGGRAPH 2011 Courses*.

BAO, Z., HONG, J.-M., TERAN, J., AND FEDKIW, R. 2007. Fracturing rigid materials. *IEEE TVCG 13*, 370–378.

BENDER, J., ERLEBEN, K., TRINKLE, J., AND COUMANS, E. 2012. Interactive simulation of rigid body dynamics in computer graphics. In *EUROGRAPHICS 2012 State of the Art Reports*.

CHADWICK, J. N., ZHENG, C., AND JAMES, D. L. 2012. Precomputed acceleration noise for improved rigid-body sound. *ACM Trans. Graph. 31*, 4, 103:1–103:9.

CHEN, Z., YAO, M., FENG, R., AND WANG, H. 2014. Physics-inspired adaptive fracture refinement. *ACM Trans. Graph. 33*, 113:1–113:7.

COUMANS, E. 2015. Bullet physics simulation. In *ACM SIGGRAPH 2015 Courses*.

ELEK, P., AND JARAMAZ, S. 2008. Fragment size distribution in dynamic fragmentation: Geometric probability approach. *FME Transactions 36*, 2, 59–65.

FRANGI, A., NOVATI, G., SPRINGHETTI, R., AND ROVIZZI, M. 2002. 3D fracture analysis by the symmetric Galerkin BEM. *Computational Mechanics 28*, 3–4, 220–232.

GLONDU, L., MUGUERCIA, L., MARCHAL, M., BOSCH, C., RUSHMEIER, H., DUMONT, G., AND DRETTAKIS, G. 2012. Example-based fractured appearance. *Computer Graphics Forum 31*, 4, 1547–1556.

GLONDU, L., MARCHAL, M., AND DUMONT, G. 2013. Real-time simulation of brittle fracture using modal analysis. *IEEE TVCG 19*, 201–209.

GRAVOUIL, A., MOËS, N., AND BELYTSCHKO, T. 2002. Non-planar 3D crack growth by the extended finite element and level sets – part II: level set update. *INT J NUMER METH ENG 53*, 11, 2569–2586.

GROSS, D., AND SEELIG, T. 2011. *Fracture Mechanics*, 2nd ed. Springer.

HAHN, D., AND WOJTAN, C. 2015. High-resolution brittle fracture simulation with boundary elements. *ACM Trans. Graph. 34*, 4, 151:1–151:12.

HERTZ, H. 1881. Über die Berührung fester elastischer Körper. *J. reine und angewandte Math. 92*, 156–171.

HIROTA, K., TANOUE, Y., AND KANEKO, T. 2000. Simulation of three-dimensional cracks. *The Visual Computer 16*, 371–378.

IBEN, H. N., AND O'BRIEN, J. F. 2006. Generating surface crack patterns. In *Proc. ACM SIGGRAPH/Eurographics SCA 2006*, 177–185.

INGRAFFEA, A. R., AND WAWRZYNEK, P. A. 2003. *Finite Element Methods for Linear Elastic Fracture Mechanics, Chapter 3.1 in Comprehensive Structural Integrity*. Elsevier Science Ltd.

JAMES, D. L., AND PAI, D. K. 1999. Artdefo: Accurate real time deformable objects. In *SIGGRAPH 99, Annual Conference Series*, 65–72.

KAUFMANN, P., MARTIN, S., BOTSCH, M., GRINSPUN, E., AND GROSS, M. 2009. Enrichment textures for detailed cutting of shells. *ACM Trans. Graph. 28*, 50:1–50:10.

KEELER, T., AND BRIDSON, R. 2014. Ocean waves animation using boundary integral equations and explicit mesh tracking. In *Proc. ACM SIGGRAPH/Eurographics SCA 2014*, 11–19.

KIELHORN, L. 2009. *A time-domain symmetric Galerkin BEM for viscoelastodynamics*. Verl. der Techn. Univ. Graz.

KOSCHIER, D., LIPPONER, S., AND BENDER, J. 2014. Adaptive tetrahedral meshes for brittle fracture simulation. In *Proc. ACM SIGGRAPH/Eurographics SCA 2014*, 57–66.

LEVINE, J. A., BARGTEIL, A. W., CORSI, C., TESSENDORF, J., AND GEIST, R. 2014. A peridynamic perspective on spring-mass fracture. In *Proc. ACM SIGGRAPH/Eurographics SCA 2014*, 47–55.

MAMOU, K., AND GHORBEL, F. 2009. A simple and efficient approach for 3D mesh approximate convex decomposition. In *16th IEEE Int. Conf. Image Processing (ICIP)*, 3501–3504.

MAYBORODA, S., AND MITREA, M. 2006. *Integral Methods in Science and Engineering: Theoretical and Practical Aspects*. Birkhäuser Boston, ch. The Poisson Problem for the Lamé System on Low-dimensional Lipschitz Domains, 137–160.

MESSNER, M., AND SCHANZ, M. 2010. An accelerated symmetric time-domain boundary element formulation for elasticity. *Engineering Analysis with Boundary Elements 34*, 11, 944–955.

MOËS, N., GRAVOUIL, A., AND BELYTSCHKO, T. 2002. Non-planar 3D crack growth by the extended finite element and level sets – part I: mechanical model. *INT J NUMER METH ENG 53*, 11, 2549–2568.

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

MOUSAVI, S. E., GRINSPUN, E., AND SUKUMAR, N. 2011. Higher-order extended finite elements with harmonic enrichment functions for complex crack problems. *INT J NUMER METH ENG 86*, 4–5, 560–574.

MÜLLER, M., MCMILLAN, L., DORSEY, J., AND JAGNOW, R. 2001. Proc. real-time simulation of deformation and fracture of stiff materials. In *Eurographic Workshop on Computer Animation and Simulation*, 113–124.

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

NORTON, A., TURK, G., BACON, B., GERTH, J., AND SWEENEY, P. 1991. Animation of fracture by physical modeling. *The Visual Computer 7*, 210–219.

O'BRIEN, J. F., AND HODGINS, J. K. 1999. Graphical modeling and animation of brittle fracture. In *SIGGRAPH 99, Annual Conference Series*, 137–146.

O'BRIEN, J. F., BARGTEIL, A. W., AND HODGINS, J. K. 2002. Graphical modeling and animation of ductile fracture. *ACM Trans. Graph. 21*, 291–294.

PARKER, E. G., AND O'BRIEN, J. F. 2009. Real-time deformation and fracture in a game environment. In *Proc. ACM SIGGRAPH/Eurographics SCA 2009*, 165–175.

PATRICIO, M., AND MATTHEIJ, R. 2007. Crack propagation analysis. *CASA-report 0723*.

PAULY, M., KEISER, R., ADAMS, B., DUTRÉ, P., GROSS, M., AND GUIBAS, L. J. 2005. Meshless animation of fracturing solids. *ACM Trans. Graph. 24*, 957–964.

POPOV, V. 2010. *Contact Mechanics and Friction*. Physical Principles and Applications. Springer.

SCHVARTZMAN, S. C., AND OTADUY, M. A. 2014. Fracture animation based on high-dimensional voronoi diagrams. In *Proc. 18th ACM SIGGRAPH i3D '14*, 15–22.

SMITH, J., WITKIN, A., AND BARAFF, D. 2001. Fast and controllable simulation of the shattering of brittle objects. *Computer Graphics Forum 20*, 81–91.

STOMAKHIN, A., SCHROEDER, C., CHAI, L., TERAN, J., AND SELLE, A. 2013. A material point method for snow simulation. *ACM Trans. Graph. 32*, 102:1–102:10.

SU, J., SCHROEDER, C., AND FEDKIW, R. 2009. Energy stability and fracture for frame rate rigid body simulations. In *Proc. ACM SIGGRAPH/Eurographics SCA 2009*, 155–164.

TERZOPOULOS, D., AND FLEISCHER, K. 1988. Modeling inelastic deformation: Viscolelasticity, plasticity, fracture. *SIGGRAPH Comput. Graph. 22*, 269–278.

ZHENG, C., AND JAMES, D. L. 2010. Rigid-body fracture sound with precomputed soundbanks. *ACM Trans. Graph. 29*, 69:1–69:13.

ZHU, Y., BRIDSON, R., AND GREIF, C. 2015. Simulating rigid body fracture with surface meshes. *ACM Trans. Graph. 34*, 4, 150:1–150:11.

Appendix

Loosely following the notation of in Eq. (A.1) in [Kielhorn 2009], the Kelvin kernels for evaluating the interior stresses given surface displacements and tractions resulting from the elastostatic 3d boundary value problem are defined as:

$$\begin{aligned}
\boldsymbol{\sigma}(\mathbf{x}) &= \int_\Gamma \left[\mathbf{S}_1(\mathbf{x},\mathbf{y})\mathbf{q}(\mathbf{y}) - \mathbf{S}_2(\mathbf{x},\mathbf{y})\mathbf{u}(\mathbf{y})\right]\, d\mathbf{y},\\
S_1[i,j,k](\mathbf{x},\mathbf{y}) :=& \frac{1}{8\pi(1-\nu)r^2}\left[(1-2\nu)(\delta_{kj}\partial_i r\right.\\
&\left.+\delta_{ki}\partial_j r - \delta_{ij}\partial_k r) + 3\partial_i r\partial_j r\partial_k r\right],\\
S_2[i,j,k](\mathbf{x},\mathbf{y}) :=& \frac{E}{8\pi(1-\nu^2)r^3}\left[3\partial_{\mathbf{n}} r((1-2\nu)\delta_{ij}\partial_k r\right.\\
&+\nu\delta_{jk}\partial_i r + \nu\delta_{ik}\partial_j r - 5\partial_i r\partial_j r\partial_k r)\\
&+3n_k(1-2\nu)\partial_i r\partial_j r + n_i((1-2\nu)\delta_{jk}\\
&+3\nu\partial_j r\partial_k r) + n_j((1-2\nu)\delta_{ik} + 3\nu\partial_i r\partial_k r)\\
&\left.-n_k\delta_{ij}(1-4\nu)\right],
\end{aligned} \tag{7}$$

with $r := |\mathbf{y}-\mathbf{x}|$ and consequently $\partial_i r = \partial r/\partial y_i = (y_i - x_i)/r$, and $\partial_{\mathbf{n}} r = (\mathbf{y}-\mathbf{x})\cdot\mathbf{n}/r$. Furthermore, $\mathbf{n}$ is the surface normal at $\mathbf{y}$, and (E,ν) are Young's modulus and Poisson's ratio respectively. The contribution of the product $\mathbf{S}_1\mathbf{q}$ to σ_{ij} in index notation is $\sum_k S_1[i,j,k]\, q_k$ and analogously for $\mathbf{S}_2\mathbf{u}$, where $i,j,k \in \{1..3\}$ refer to the coordinate axes.

The entries of the system matrix blocks in Eq. (4) are defined as follows, see also Eq. (5.17) in [Kielhorn 2009], where the indices i,j enumerate nodes on the object's surface and k,l refer to nodes on fracture surfaces:

$$\begin{aligned}
\mathbf{K}_{ij} &:= \int_\Gamma \psi_i(\mathbf{x})\mathcal{T}_{\mathbf{x}} \int_\Gamma (\mathcal{T}_{\mathbf{y}}\mathbf{U})^\top(\mathbf{y}-\mathbf{x})\psi_j(\mathbf{y}) ds_{\mathbf{y}} ds_{\mathbf{x}},\\
\mathbf{C}_{ik} &:= \int_\Gamma \psi_i(\mathbf{x})\mathcal{T}_{\mathbf{x}} \int_\Gamma (\mathcal{T}_{\mathbf{y}}\mathbf{U})^\top(\mathbf{y}-\mathbf{x})\psi^c_k(\mathbf{y}) ds_{\mathbf{y}} ds_{\mathbf{x}},\\
\mathbf{D}_{lk} &:= \int_\Gamma \psi^c_l(\mathbf{x})\mathcal{T}_{\mathbf{x}} \int_\Gamma (\mathcal{T}_{\mathbf{y}}\mathbf{U})^\top(\mathbf{y}-\mathbf{x})\psi^c_k(\mathbf{y}) ds_{\mathbf{y}} ds_{\mathbf{x}}.
\end{aligned} \tag{8}$$

We use piecewise-linear shape functions, ψ, for displacements; superscript-c indicates shape functions corresponding to crack-opening degrees of freedom.

The right-hand side vector is assembled from the given surface traction field, $\mathbf{q}$:

$$\mathbf{f}_j := \int_\Gamma \psi_j(\mathbf{x}) \left[\frac{1}{2}\mathbf{q}(\mathbf{x}) + \int_\Gamma (\mathcal{T}_{\mathbf{x}}\mathbf{U})(\mathbf{y}-\mathbf{x})\mathbf{q}(\mathbf{y}) ds_{\mathbf{y}}\right] ds_{\mathbf{x}}. \tag{9}$$

Finally, $\mathbf{U}$ is the 3d elastostatic fundamental solution (or Green's function), see Eq. (4.46) in [Kielhorn 2009], and the traction operator $\mathcal{T}$ is a conormal derivative and defined according to Eq. (1.4) in [Mayboroda and Mitrea 2006] as: $\mathcal{T}_{\mathbf{x}}\mathbf{u} := \lambda(\nabla_{\mathbf{x}}\cdot\mathbf{u})\mathbf{n}_{\mathbf{x}} + \mu(\nabla_{\mathbf{x}}\mathbf{u} + (\nabla_{\mathbf{x}}\mathbf{u})^\top)\mathbf{n}_{\mathbf{x}}$, where $\mathbf{n}_{\mathbf{x}}$ is the outward unit surface normal at $\mathbf{x}$ and (λ,μ) are Lamé parameters.

Source code is available at
http://pub.ist.ac.at/group_wojtan/projects/2016_Hahn_FastFracture.