Simulating Rigid Body Fracture with Surface Meshes

Yufeng Zhu*
University of British Columbia

Robert Bridson†
Autodesk, University of British Columbia

Chen Greif‡
University of British Columbia

Abstract

We present a new brittle fracture simulation method based on a boundary integral formulation of elasticity and recent explicit surface mesh evolution algorithms. Unlike prior physically-based simulations in graphics, this avoids the need for volumetric sampling and calculations, which aren't reflected in the rendered output. We represent each quasi-rigid body by a closed triangle mesh of its boundary, on which we solve quasi-static linear elasticity via boundary integrals in response to boundary conditions and loads such as impact forces and gravity. A fracture condition based on maximum tensile stress is subsequently evaluated at mesh vertices, while crack initiation and propagation are formulated as an interface tracking procedure in material space. Existing explicit mesh tracking methods are modified to support evolving cracks directly in the triangle mesh representation, giving highly detailed fractures with sharp features, independent of any volumetric sampling (unlike tetrahedral mesh or level set approaches); the triangle mesh representation also allows simple integration into rigid body engines. We also give details on our well-conditioned integral equation treatment solved with a kernel-independent Fast Multipole Method for linear time summation. Various brittle fracture scenarios demonstrate the efficacy and robustness of our new method.

CR Categories: G.1.9 [Mathematics of Computing]: Numerical Analysis—Integral Equations; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

Keywords: dynamics, fracture, rigid body, boundary integrals, triangle mesh, Fast Multipole Method

1 Introduction

High-quality rigid body fracture has been a popular research topic in computer graphics, with the purpose of increasing realism and plausibility in computer-generated animation. Fracturing effects such as shattering glass, destruction of a cement wall or blowing up a space battleship greatly affect the immersive user experience in computer games, virtual reality and film industry. Roughly speaking, current state-of-the-art approaches can be categorized into two groups: (1) crack propagation and (2) volume decomposition. The first category uses a volume discretization to solve for the underlying elastic dynamics, and applies some crack growth criteria, such as principal stress hypothesis (Rankine hypothesis), critical strain energy release rate (Griffith criterion) and minimum strain energy density (Beltrami hypothesis), to propagate fractures inside the material. These methods are generally physically realistic, but entail significant computational cost due to either the poor scaling behavior of the physical system or the complexity of the unstructured volumetric grid's remeshing operation. The second category starts with spatial decomposition of the object instantly, instead of waiting for the crack propagation process to create new fractured pieces. Such a simplified strategy gives rise to computational efficiency and flexible control of fragment generation, which is essential and attractive to graphics experts, practitioners and users. However, the potentially unpredictable physically inconsistent behavior caused by this simplification might require significant post-processing.

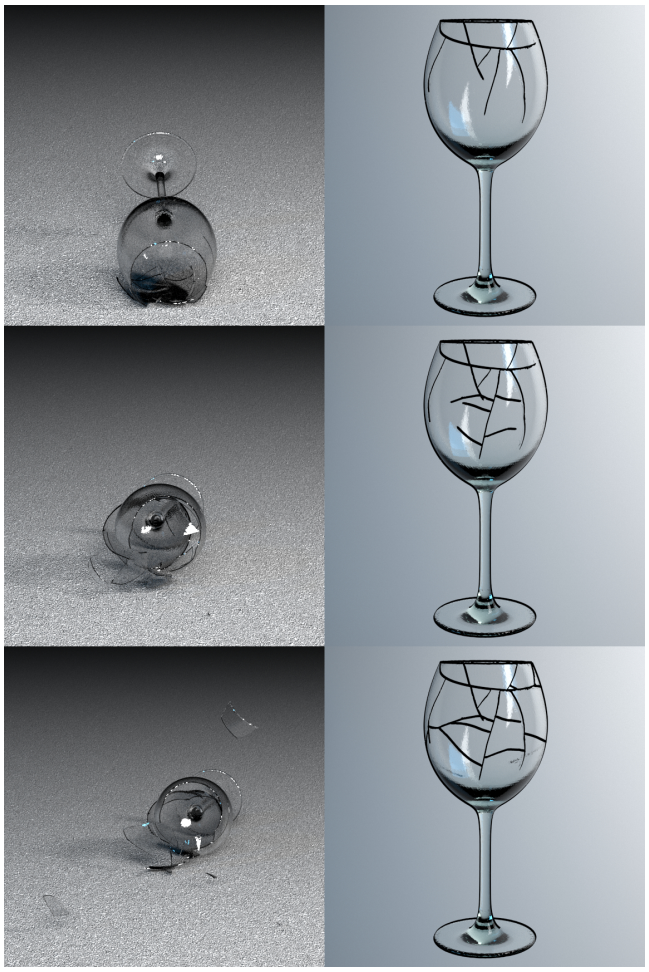


Figure 1: ***A glass goblet*** *broken into multiple shards (**left**), with corresponding fracture propagation shown in the material space (**right**). Only the surface mesh is used for both the physical computation and fracture propagation.*

Our goal is to design a new rigid body fracture algorithm that is physically plausible and computationally efficient. We present a surface-only method to handle both physical computation and fracture evolution by formulating the quasi-static elasticity problem as

*e-mail:mike323@cs.ubc.ca
†e-mail:rbridson@cs.ubc.ca
‡e-mail:greif@cs.ubc.ca

an indirect boundary integral equation and using explicit mesh-based interface tracking technique (see Figure 1). By making use of the indirect formulation, we eliminate the non-trivial null space in the system of free-flying objects, which is an inherent problem for the direct formulation. Elastic stress tensors, which are the crucial ingredient for our fracture initiation and fracture propagation method, can be thus written in a consistent analytical form. We initiate and propagate crack interface in the spirit of the Rankine hypothesis. Each crack front will be generated and evolved under the principal stress flow in the object's material space explicitly and independently. We use existing robust mesh-based surface tracking methods, but incorporate significant modifications in order to adapt the tracking algorithm to our fracture evolution problem. The explicit approach not only avoids the need for a signed distance volume grid but also preserves tiny surface geometrical features. Another benefit is that our method can be easily combined with any popular rigid body dynamics engine by using an explicit surface representation. In our implementation, we use a velocity level linear complementary programming (LCP) rigid body solver. The physical equation using our indirect formulation is well-conditioned, and discretization of the integral equation leads to a dense system, which is small relative to the the sparse system that arises in the volumetric approach, but it nonetheless requires a sophisticated solution approach. To that end, we modify a kernel-independent Fast Multipole Method and combine it with a BiCG-STAB iterative solver, to obtain a linear runtime in the number of surface triangles.

In summary, our core technical contributions are as follows:

- we have developed the first rigid body fracturing physical simulation that only requires surface representation and surface mesh operations;
- we have designed and implemented a new well-conditioned indirect boundary integral formulation for the linear quasi-static elasticity problem;
- we offer the first explicit mesh-based surface tracking algorithm for the rigid body brittle fracture problem.

We also highlight the flexibility of the kernel-independent Fast Multipole Method designed for the linear time n-body fast summation, which to the best of our knowledge has not been previously used in computer graphics models and applications.

2 Related Work

Fracture Simulation. Since the pioneering work of Terzopoulos and Fleischer [1988] and Norton et al. [1991], fracture modeling has been an important part of physically based animation and interactive virtual environment. In their seminal work, O'Brien and his collaborators [O'Brien and Hodgins 1999; O'Brien et al. 2002] simulated brittle and ductile fracture by making use of the finite element method (FEM). In order to deal with the elastodynamic problem, researchers used a corotational formulation to handle large deformation and fracturing effects [Müller and Gross 2004; Irving et al. 2004; Parker and O'Brien 2009]. One of the most challenging problems in FEM-based fracture simulation arises from the topology change of the volumetric mesh. Mesh elements cut by the crack front must be dynamically remeshed so as to have the simulation domain conform with the fracture surface [O'Brien and Hodgins 1999; Wicke et al. 2010]. A naïve treatment, like deleting elements or aligning the crack plane with the elements' boundary, may lead to a jagged result. Molino et al. [2004] and Sifakis et al. [2007] introduced a virtual node algorithm to decouple material domain and simulation domain by duplicating elements necessary for modeling the topological change. To avoid volumetric mesh updates, Pauly et al. [2005] and Steinemann et al. [2006] developed meshless fracturing methods as a composition of mesh free simulation and crack front tracking. Unlike these fracture propagation approaches based on surface reconstruction or crack synthesis, we develop a new crack front tracking algorithm using simple triangle mesh operations similar to explicit mesh-based surface tracking techniques [Brochu and Bridson 2009; Da et al. 2014]. Moreover, by formulating the linear quasi-static stress analysis [Müller et al. 2001; Bao et al. 2007; Zheng and James 2010] in our indirect boundary integral formulation, we place all degrees of freedom on the surface only and get rid of the visibility graph as required by previous meshless methods. Other interesting fracturing work includes thin sheet problem [Kaufmann et al. 2009; Busaryev et al. 2013; Pfaff et al. 2014], Voronoi diagram methods [Bao et al. 2007; Su et al. 2009; Müller et al. 2013] and material strength field synthesis [Chen et al. 2014].

Boundary Integral Equations. Boundary integral equations (BIE) are reformulations of boundary value problems for partial differential equations (PDEs). They have been present in the mathematics literature for almost two centuries, and engineers have made much use of them for solving a wide range of real-world problems. Their promise, motivating the present work, is to significantly reduce the number of degrees of freedom compared to volumetric methods, and simplify the geometric part of the problem to surface meshes only. For example, to solve elasticity in a region with an n-element surface mesh, a BIE solver accelerated by FMM or similar strategies takes $O(n)$ time, and needs nothing more than the surface mesh. By comparison, a uniform tetrahedral mesh conforming to the surface would contain $O(n^{1.5})$ elements, and the common solver choice of a sparse Cholesky factorization would take $O(n^2)$ space and $O(n^3)$ time.

The use of BIE methods in physical animation dates back to the work of James and Pai [1999], who formulated the elastostatic problem as direct boundary integral. Such a formulation is suitable for elliptic PDEs with Dirichlet boundary conditions, but it has a nontrivial null space with pure Neumann boundary condition, for example a free flying rigid object. Researchers addressed this problem within the FEM framework and overcame it by anchoring degrees of freedom or projecting out the null space algebraically [Müller et al. 2001; Bao et al. 2007; Zheng and James 2010]. Our own approach is to tackle this ill-conditioned PDE by exploring an indirect boundary integral formulation which has previously been applied to vortex sheets [Brochu et al. 2012] and ocean waves [Keeler and Bridson 2014]. A primary characteristic of our method is that it involves solving a second kind Fredholm integral equation, which is guaranteed to be well-conditioned. More details are provided in § 4.2. Other graphics applications of BIE include the vortex method [Park and Kim 2005; Weissmann and Pinkall 2010; Golas et al. 2012], time harmonic sound rendering [James et al. 2006] and diffusion curve imaging [Sun et al. 2014]. BIE has also been widely explored in mechanic field, such as elstostatic problem, elastodynamics and fracture analysis, using various discretization approaches including collocation scheme [Blandford et al. 1981], Bubnov Galerkin [Frangi 2002], Petrov Galerkin [Sladek et al. 2004], etc. Approaches considering elastic wave include time-domain boundary element formulation [Messner and Schanz 2010], dual reciprocity method [Portela et al. 1992] as well as time-harmonic solution [Rezayat et al. 1986]. Since dynamic approach involves special treatment on crack tip during fracture propagation to keep numerical accuracy [Ang 2014], we instead look at the problem from a quasistatic point of view. We also refer interested readers to [Chunrungsikul 2001] for more discussions on singular integral quadrature.

Crack Front Tracking. Conventional FEM-based Lagrangian fracture methods use volumetric mesh cutting algorithms to keep track of fracture propagation. See [Wu et al. 2014] for a compre-

Algorithm 1 Rigid Body Fracture Simulation Loop

solve rigid body dynamics [fig. 2a]
estimate rigid body contact force [fig. 2b]
solve layer potential for elastostatic problem [fig. 2c]
apply stress analysis on surface nodes [fig. 2d]
add local maximum nodes to fracture node list $\mathcal{N}$
set fracture line list $\mathcal{L} \leftarrow$*Crack_Initiation*($\mathcal{N}$)
for each line l **in** $\mathcal{L}$
    label all nodes, except the ends, of l as active
end for
while $\mathcal{L} \neq \emptyset$ **do**
    apply stress analysis on active crack nodes of $\mathcal{L}$ [fig. 2d]
    label nodes as inactive if they fail on Rankine condition
    assign propagation velocity to active crack nodes [eq. (9)]
    Crack_Propagation($\mathcal{L}$, sub_timestep) [fig. 2e]
end while
update rigid body list if new fragments are created

hensive and detailed review of previous cutting algorithms in the literature. Our fracture propagation algorithm is mostly related to the mesh-based surface tracking techniques [Brochu and Bridson 2009; Da et al. 2014] which have been successfully applied to multiphase flow. We make several essential modifications to the mesh operation strategy of the *LosTopos* library [Da et al. 2014] in order to resolve the particular physical behaviour of crack initiation and propagation. Our mesh operation strategy does depend on the crack opening criterion. We use the Rankine condition, because it seems to be much more effective than the Griffith energy type approach [Hegemann et al. 2013] in preserving volume. The Rankine approach provides us with the flexibility to control crack open thickness, and it yields a physically convincing simulation result.

Fast Integral Equation Solver. Compared with FEM or the finite difference method (FDM) for PDEs, the boundary integral formulation has the advantage of having fewer degrees of freedom. However, the fact that unknown variables in BIE are all coupled leads to dense system after discretization, which hinders the direct application of a powerful iterative linear system solver. By using the low-rank property of n-body interaction with non-oscillatory kernel, algorithms of O(N) or O(N$\log$ N) complexity for matrix-vector multiplication have been developed since 1980's, including particle-particle/particle-mesh (P^3M) [Hockney and Eastwood 1988], tree code [Barnes and Hut 1986], panel clustering [Hackbusch and Nowak 1989] and fast multipole method (FMM) [Greengard and Rokhlin 1987]. A second class of methods are wavelet compression techniques [Alpert et al. 1993], which lead to sparse and asymptotically well-conditioned approximations of the coefficient matrix. Applications of the fast iterative dense system solver in computer graphics include deformable material [James and Pai 2003], radiosity rendering [Hanrahan et al. 1991], fluid simulation [Brochu et al. 2012; Keeler and Bridson 2014] and sound rendering [Zheng and James 2010]. Instead of applying a spherical multipole expansion to the kernel function which is a second order tensor in the elastostatic case, we introduce a radial basis function (RBF) style based kernel independent FMM in § 5 by simplifying the implementation of Ying et al. [2004]. By utilizing the RBF field approximation, we are able to write the multipole expansion, multipole to multipole (M2M), multipole to local (M2L), local to local (L2L) and local expansion in a simple consistent form.

3 Algorithm Overview

Our rigid body fracture simulation method is closely related to the work by Müller et al. [2001], Bao et al. [2007] and Zheng and James [2010], who also applied a quasi-static linear elastic model to stress analysis. We use an LCP-based rigid body simulator [Erleben 2007] for the purpose of stability and simplicity. We are not discussing rigid body simulation in this paper but refer interested readers to [Bender et al. 2012] for a comprehensive review. Figure 2 provides a brief illustration of the work flow in our fracturing system, while detailed major steps in each timestep iteration are summarized in Algorithm 1. Each iteration starts with a rigid body simulation, and contact forces are computed for each object. If there are rigid bodies in contact, the contact forces and the gravitational forces will be transformed into the material space, respectively, and used as Neumann boundary conditions and inhomogeneous terms for our boundary integral equation. The dense system generated by such integral equation will be solved using the BiCGSTAB iterative solver [van der Vorst 1992], combined with a fast summation method. Once we obtain the layer potential, we can evaluate elastic deformation, like displacement and stress, in the material domain. Such information will be used to start and propagate fracture by checking the Rankine condition. Finally, after all fractures finish propagating or cut completely through the material, we update the rigid body list if there are new fragments generated by fracturing process.

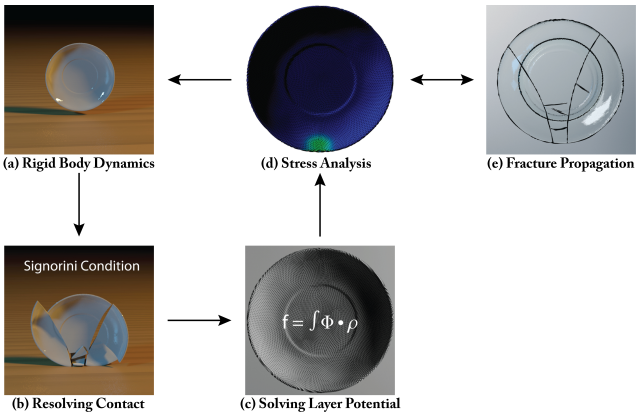


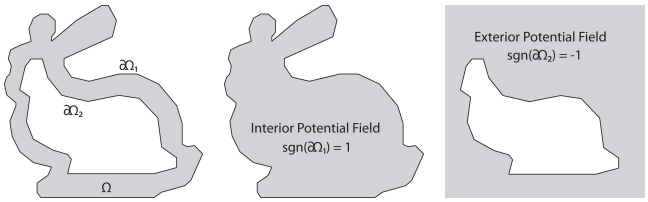
Figure 2: *A graphical visualization of* ***Algorithm*** 1*: our rigid body simulation loop is composed of* ***(a)*** *Rigid Body Dynamics,* ***(b)*** *Resolving Contact,* ***(c)*** *Solving Layer Potential,* ***(d)*** *Stress Analysis and* ***(e)*** *Fracture Propagation.*

Due to the characteristics of rigid bodies, linear elasticity is suitable for simulating material undergoing small deformations. Even though the quasi-static model doesn't capture elastic wave propagation, which affects fracture as noted by Glondu et al. [2013], our indirect boundary integral can be extended to elastodynamic model similarly to acoustic scattering models [James et al. 2006]. Such an extension could be easily combined with our other modules.

4 Boundary Element Simulation

4.1 Elastostatic Constitutive Model

We assume that the material to be simulated always satisfies hyperelastic and isotropic properties. Traditional approaches involve defining the elastic constitutive relation from an energy density and using the first Piola Kirchhoff stress tensor, $\mathbf{P}$. However, for the boundary element method used in this paper, it is necessary to build the model upon the second Piola Kirchhoff stress tensor, $\mathbf{S}$. By so doing, we obtain a linearized mapping between the displacement $\mathbf{u}(\mathbf{x})$ and the stress tensor $\mathbf{S}(\mathbf{x})$, with a small deformation assump-



(a) **(b)** **(c)**

Figure 3: ***(a)*** *A general material domain enclosed by two codimension one embedded submanifolds (lines);* ***(b)*** *When the submanifold is exterior to the enclosed domain, it constructs an interior potential field whose jumping condition on it is positive;* ***(c)*** *Otherwise, it constructs an exterior potential field with a negative jumping condition. The surface normal is consistent and always pointing out of the material domain.*

tion:

$$\begin{aligned} \mathbf{F}(\mathbf{x}) &= \nabla \mathbf{u}(\mathbf{x}) + \mathbf{I}, \quad \mathbf{x} \in \Omega \\ \boldsymbol{\varepsilon}(\mathbf{x}) &= \frac{1}{2}(\mathbf{F}(\mathbf{x}) + \mathbf{F}^T(\mathbf{x})) - \mathbf{I}, \quad \mathbf{x} \in \Omega \\ \mathbf{S}(\mathbf{x}) &= \mathcal{C} : \boldsymbol{\varepsilon}(\mathbf{x}), \quad \mathbf{x} \in \Omega \end{aligned} \tag{1}$$

Here $\mathbf{F}$ is the deformation gradient and Ω represents the material space. $\boldsymbol{\varepsilon}$ is a linearized right Cauchy-Green strain tensor, and $\mathcal{C}$ denotes a symmetric fourth order material elasticity tensor. The specific linear elasticity model we use is

$$\mathbf{S} = 2\mu\boldsymbol{\varepsilon} + \lambda \mathbf{tr}(\boldsymbol{\varepsilon})\mathbf{I},$$

where λ, μ are $Lam\acute{e}$ parameters.

Similar to previous approaches [Bao et al. 2007; Su et al. 2009; Müller et al. 2013], we adopt a quasi-static physical model in the stress analysis, by ignoring the influence of the elastic waves on fracture propagation. That is, there are no inertia terms, and we obtain the simplified equations

$$\begin{aligned} \nabla \cdot \boldsymbol{\sigma}(\mathbf{x}) &= \mathbf{g}, \quad \mathbf{x} \in \Omega \\ \boldsymbol{\sigma}(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x}) &= \mathbf{f}(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega, \end{aligned} \tag{2}$$

where $\mathbf{g}$ and $\mathbf{f}$ represent body forces such as gravity and boundary traction or contact force, respectively. Notice that $\boldsymbol{\sigma}$ represents the Cauchy stress tensor, which is related to $\mathbf{S}$ via

$$\boldsymbol{\sigma} = \frac{1}{\det(\mathbf{F})}\mathbf{F}\mathbf{S}\mathbf{F}^T. \tag{3}$$

Even though in general the stress tensors $\boldsymbol{\sigma}$ and $\boldsymbol{S}$ are different, it can be shown that they are equivalent under geometrical linearization when the deformation is small. By substituting $\boldsymbol{\sigma}$ into Equation 1, we actually obtain the *Navier-Cauchy* equation, which is solved using a direct boundary integral formulation in [James and Pai 1999] to accomplish interactive haptic rendering.

4.2 Indirect Boundary Integral

We now explore the elliptic characteristics of *Navier-Cauchy* equation and transform the PDE into an indirect boundary integral equation using potential theory. A similar idea has been applied to potential flow [Brochu et al. 2012; Keeler and Bridson 2014] in computer graphics to solve Laplace's equation. In our case, we are dealing with a Poisson-style equation concerning the inhomogeneous term due to $\mathbf{g}$. We represent the displacement solution $\mathbf{u}(\mathbf{x})$, when $\mathbf{x}$ lies in the material domain Ω enclosed by m codimension one embedded submanifolds $\partial\Omega_p$, as shown in Figure 3, as a layer potential with an additional potential term to account for the body force:

$$\begin{aligned} \mathbf{u}(\mathbf{x}) = &\sum_{p=0}^{m} \oint_{\partial\Omega_p} \boldsymbol{\Phi}(\|\mathbf{x}-\mathbf{y}\|_2)\boldsymbol{\rho}(\mathbf{y})d\mathbf{y} \\ &+ \oint_{\partial\Omega_p} \boldsymbol{\aleph}(\|\mathbf{x}-\mathbf{y}\|_2) \otimes_2 \mathbf{g} \otimes_3 \mathbf{n}_\mathbf{y} d\mathbf{y}, \quad \mathbf{x} \in \Omega \end{aligned} \tag{4}$$

where the first summand is a single layer potential term while the second is a Newton potential term [Meßner 2008]. Such a representation only handles Neumann boundary conditions, as we focus on free-flying rigid body fracturing in this project. A complete solution could be achieved by incorporating a double layer potential to account for Dirichlet boundary conditions which appear frequently in articulated body simulation. The operator $\otimes_n$ is a conventional n-mode tensor vector multiplication, defined as follows using Einstein notation:

$$\begin{aligned} \mathcal{M} \otimes_n \mathbf{v} &= \mathcal{M}_{I_1 I_2 \cdots I_n \cdots I_N} \mathbf{v}_{I_n} \\ \mathcal{M} \in \mathbb{R}^{I_1 \times I_2 \times \cdots I_n \cdots \times I_N}&, \quad \mathbf{v} \in \mathbb{R}^{I_n \times 1} \end{aligned} \tag{5}$$

$\boldsymbol{\Phi}(r)$ and $\boldsymbol{\aleph}(r)$ are, respectively, the fundamental solution and a higher order fundamental solution's gradient of the *Navier-Cauchy* equation:

$$\begin{aligned} \boldsymbol{\Phi}_{ij}(r) &= \frac{1}{8\pi\mu r}\left[2\delta_{ij} - \left(\frac{\lambda+\mu}{\lambda+2\mu}\right)(\delta_{ij} - r_{,i}r_{,j})\right] \\ \boldsymbol{\aleph}_{ijk}(r) &= \frac{r_{,k}}{8\pi\mu} - \frac{1}{32\pi\mu}\left(\frac{\lambda+\mu}{\lambda+2\mu}\right)\chi_{ijk}(r) \end{aligned} \tag{6}$$

where

$$\chi_{ijk}(r) = \delta_{ij}r_{,k} + \delta_{ik}r_{,j} + \delta_{jk}r_{,i} + r_{,i}r_{,j}r_{,k}.$$

δ is the Kronecker delta. $\boldsymbol{\rho}(\mathbf{y})$ denotes the charge density of layer potential in potential theory and it is the unknown variable in our formulation. To solve for $\boldsymbol{\rho}(\mathbf{y})$, we use Equation 2 combined with Equation 1 and Equation 3, which yields

$$\begin{aligned} \mathbf{f}(\mathbf{x}) = &\kappa(\mathbf{x})\boldsymbol{\rho}(\mathbf{x}) + \sum_{p=0}^{m} \oint_{\partial\Omega_p} \frac{\partial \boldsymbol{\aleph}(\|\mathbf{x}-\mathbf{y}\|_2)}{\partial \mathbf{n}_\mathbf{x}} \otimes_2 \mathbf{g} \otimes_3 \mathbf{n}_\mathbf{y} d\mathbf{y} \\ &+ \frac{1}{2}\oint_{\partial\Omega_p} \mathcal{C} : \Big(\Gamma(\mathbf{x},\mathbf{y}) + \Gamma^T(\mathbf{x},\mathbf{y})\Big) \cdot \mathbf{n}_\mathbf{x} d\mathbf{y}, \quad \mathbf{x} \in \partial\Omega_q \end{aligned} \tag{7}$$

where

$$\begin{aligned} \kappa(\mathbf{x}) &= \mathrm{sgn}(\partial\Omega_q)\frac{\omega(\mathbf{x})}{4\pi}, \quad \mathbf{x} \in \partial\Omega_q \\ \Gamma(\mathbf{x},\mathbf{y}) &= \nabla_\mathbf{x}\boldsymbol{\Phi}(\|\mathbf{x}-\mathbf{y}\|_2)\rho(\mathbf{y}), \quad \mathbf{x} \in \partial\Omega_q,\ \mathbf{y} \in \partial\Omega_p. \end{aligned}$$

The quantity $\kappa(\mathbf{x})\rho(\mathbf{x})$ accounts for the jump behavior of the second integral operator when $\mathbf{x}$ approaches the domain's boundary; $\mathrm{sgn}(\partial\Omega)$ is a sign function which equals 1 when $\partial\Omega$ encloses an interior domain and -1 otherwise; and $\omega(\mathbf{x})$ is a solid angle at the node $\mathbf{x}$. Note that both $\mathbf{f}(\mathbf{x})$ and $\mathbf{g}(\mathbf{x})$ are represented in the material space. For a more detailed derivation of the layer potential model we presented in this section, please refer to Section 1 of the supplemental material.

4.3 Boundary Element Discretization

In general the solution of an elliptic PDE can be represented as a single layer potential, double layer potential, or even a mixed potential. Single layer potentials give rise to Fredholm equations of

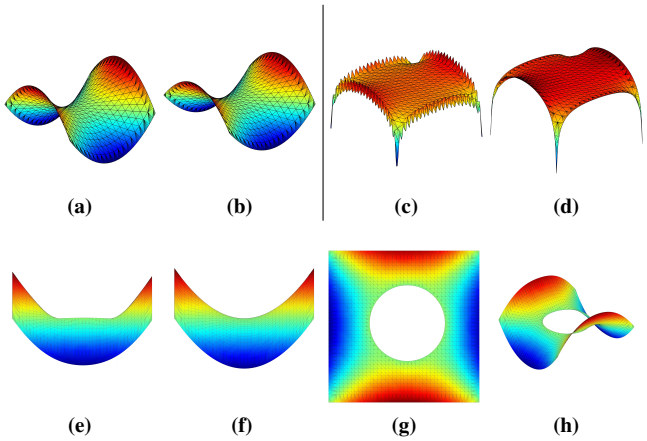


Figure 4: ***Upper:*** *We compare our formulation and another online implementation by solving the same problems with compatible boundary conditions in* ***a*** *and* ***b*** *(ours) as well as with incompatible boundary condition in* ***c*** *and* ***d*** *(ours);* ***Lower:*** *Problems with more complicated domains* ***e****,* ***g****,* ***h*** *with zero Neumann boundary conditions on the circle,* 1 *on one pair of square sides and* -1 *on the other pair. A solution for a simple square domain* ***f*** *is also included, for comparison.*

the second kind after taking the normal derivative, which leads to a well-conditioned linear system for Neumann boundary problem after discretization. Such well-conditioning property is guaranteed by the *Fredholm Alternative* theory from functional analysis. In Figure 4 (**a**, **b**, **c**, **d**), we compare our implementation with another mixed potential method [Kirkup and Yazdani 2008] for 2D Laplace's problem. Both methods achieve similar results for compatible boundary conditions; see (**a**, **b**). On the other hand, our method generates a smoother solution in (**d**), as opposed to (**c**), for incompatible boundary conditions. We also show that our approach can be easily applied to more complicated domains in (**e**, **f**, **g**, **h**).

We discretize Equation 7 using piecewise constant shape functions, and adopt a collocation scheme at triangle centroids:

$$(\mathbf{D}+\mathbf{A})\boldsymbol{\rho} = \mathbf{f} - \tilde{\mathbf{g}} \tag{8}$$

where $\mathbf{D}$ is diagonal with nonzero entries corresponding to $\kappa(\mathbf{x})\rho(\mathbf{x})$, and $\mathbf{f}$ collects all the discretized boundary tractions. $\mathbf{A}$ is a dense matrix corresponding to the second summand on the right-hand side of Equation 7, while $\tilde{\mathbf{g}}$ is composed of the first summand. Please refer to Section 1.4 of the supplemental material for assembly details of Equation 8. After obtaining the layer potential $\boldsymbol{\rho}$ by solving the above equation, we can evaluate the deformation displacement using Equation 4 and elastic stress by substituting the displacement into Equation 1 at any point in the material. 3D elastic results with displacements evaluated on the boundary surface and volume energy visualization are shown in Table 1. Other discretization approaches, including the Galerkin projection method, *Nyström* method combined with high order quadrature rules like quadrature by expansion (QBX) [Köckner et al. 2013], can also be applied to obtain higher order convergence rates.

During the fracturing process, we need to run stress analysis at candidate nodes, either for the purpose of opening a crack or propagating a crack. We evaluate the elastic stress at these positions and compute the eigendecomposition of it directly, due to the quasi-static elasticity assumption. The maximum positive eigenvalue is used to decide whether a candidate node should undergo material failure, or its fracture propagation velocity if it is already a crack node.

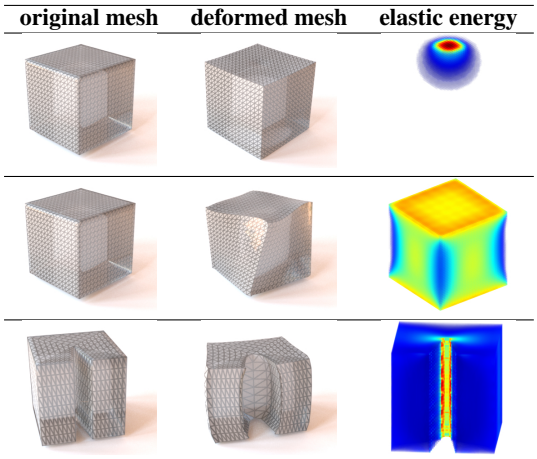


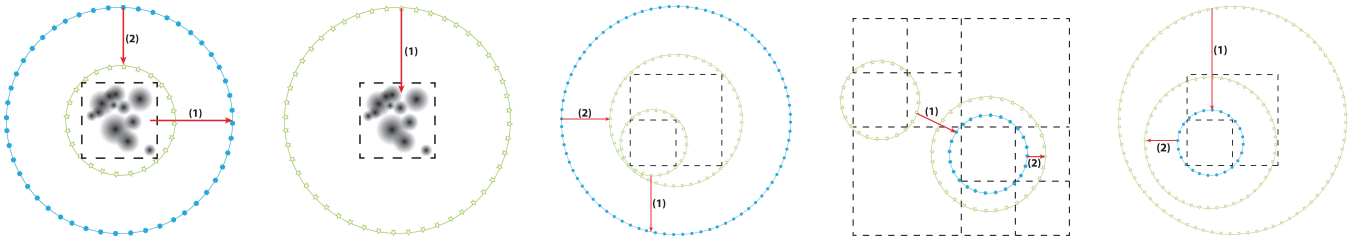
Table 1: *Elastic deformation solved by the indirect boundary integral formulation.* ***first row:*** *applying a point force load on the top face;* ***second row:*** *twisting the cube;* ***third row:*** *stretching the object.*

5 Fast Multipole Method

Although the sum of $\mathbf{D}$ and $\mathbf{A}$ in Equation 8 forms a well-conditioned matrix, its dense structure hinders the application of powerful iterative linear system solvers. By making use of a fast summation method like FMM, we can achieve linear running time to solve the dense system; see, for example, the work by Sun et al. [2014] for a clear introduction to 2D FMM. The same framework can be extended to the 3D version by replacing the Laurent expansion with a spherical multipole expansion of the kernel function [Greengard and Rokhlin 1987]. Instead of applying such a complicated expansion to our second order tensor kernel function, we introduce a simplified kernel-independent FMM which only requires kernel evaluation. Our work is closely related to [Ying et al. 2004]. Instead of using a layer potential as they did, we make use of an RBF-style method to approximate far-field interactions. More specifically, for boundary integrals with different integrands (kernel functions), we choose corresponding kernel function as the RBF basis for our FMM, so there is no need for either kernel expansions or extra layer potential evaluations.

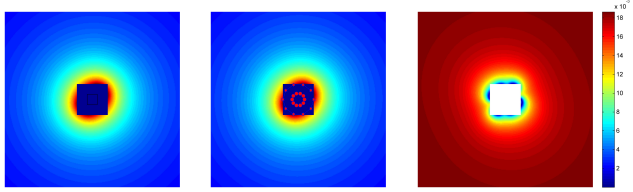
Given a cluster of particles lying on a smooth manifold, our kernel-independent FMM starts with building an octree spatial partition. For each octree node that has a non-empty interaction list, we assign two virtual bounding spheres as illustrated in Figure 5a. The virtual sphere is identified as either a *representing sphere* or a *checking sphere*. The RBF basis is placed on the exterior sphere to approximate the influence from far field to local, while the interior sphere is used to approximate the influence from local to far field. To determine the RBF's coefficients $\boldsymbol{\alpha}$, we evaluate the values $\boldsymbol{\eta}$ on the checking sphere (the first step in Figure 5**a**, **b**, **c**, **d** and **e**) and solve a linear system $\mathbf{K}\boldsymbol{\alpha} = \boldsymbol{\eta}$ (the second step in Figure 5**a**, **c**, **d** and **e**). As opposed to the Tikhonov regularization solver used by [Ying et al. 2004], we observe that our matrix $\mathbf{K}$ is well-conditioned, and therefore we solve the system directly by computing the LU decomposition of the matrix. By fixing the size of both virtual spheres relative to the size of the octree node, we pre-compute the LU factorization for a cubic box with unit length once at the beginning and apply it with a sizing multiplier ϖ during run time. ϖ is the scaling factor of octree node relative to the unit box. We present the RBF approximation of a 2D problem in Figure 6.

Finally, we follow the conventional FMM scheme and complete our



(a) *Multipole Expansion* **(b)** *Local Expansion* **(c)** *Multipole to Multipole* **(d)** *Multipole to Local* **(e)** *Local to Local*

Figure 5: *A 2D illustration of RBF based (**a**) multipole expansion (**b**) local expansion (**c**) multipole to multipole translation (**d**) multipole to local expansion (**e**) local to local translation. Circles in blue are checking circle while circles in green are representing circle. The red arrows point from known data to the unknowns we want to solve. Numbers on the arrows represent the order of computation.*



(a) *Exact Potential* **(b)** *RBF Potential* **(c)** *Relative Error*

Figure 6: *An illustration of the use of an RBF method to approximate a 2D anisotropic exterior potential field: (**a**) exact far field generated by 1**e**5 particles with random mass placed nonuniformly inside the center black box (**b**) approximated far field by only 10 degrees of freedom placed at red dots (representing circle) and red stars are checking circle (**c**) relative error between exact field and approximated field*

fast summation in three steps. First, we traverse the octree structure in a bottom-up order, applying multipole expansions to leaf nodes as well as multipole-to-multipole translations to each node (Figure 5a, c). The expansion results are stored as RBF coefficient vectors for each traversed node. Next, a multipole-to-local expansion step is taken between each tree node and nodes in its interaction list as shown by Figure 5d. We finish the summation by traversing top-down through the octree, aggregating the far-field influences by applying local-to-local translations to each node (Figure 5e) as well as an additional composition step to each leaf node. In the composition step, we use a local expansion to account for far-field influences (Figure 5b) and a direct summation for the influence from near-field particles. Please see Section 2 of the supplemental material for a thorough discussion of the implementation details. A runtime comparison between our FMM and brute force summation is provided; see Table 2 and Figure 7.

By making use of FMM, we can compute matrix-vector products in linear time. To solve Equation 8, we combine FMM with the BiCGSTAB iterative solver and achieve fast convergence as well as an accurate solution. More detailed numerical results are provided in § 7.

Model	Vertices	FMM(s)	BF(s)	Error(L_∞)
star	2675	5.639	0.095	1.1e-5
armadillo	23245	6.522	7.071	2.15e-4
fertility	220332	56.642	648.971	7.7e-5
moebius	704480	273.339	6577.087	4.3e-5
dragon	3609600	949.959	206429.02	6.1e-5

Table 2: *Detailed input information, algorithm timing and accuracy for tests shown in Figure 7 run (sequentially) on a desktop with an Intel i7-3770K and 16GB of memory.*

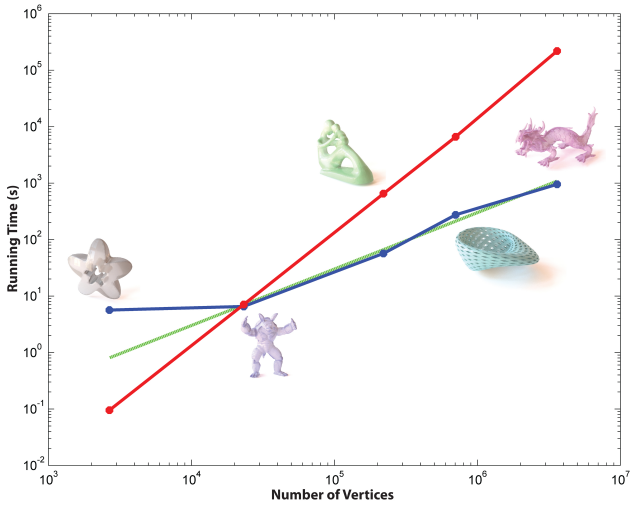


Figure 7: *Runtime comparison between an $O(n^2)$ brute force method (red), $O(n)$ complexity (green dash) and our FMM (blue) using the 3D Navier-Cauchy equation's fundamental solution as an RBF basis.*

6 Fracture Propagation Tracking

Traditional physically-based fracture simulation generates and evolves cracks by cutting through volumetric mesh elements. Cutting algorithms have been developed for different kinds of elements, including tetrahedra, polyhedra and point clouds [Wu et al. 2014]. To avoid volumetric mesh cutting and remeshing, meshless methods have been proposed to track crack evolution in the material space directly with an additional surface reconstruction or constructive solid geometry (CSG) step [Pauly et al. 2005; Steinemann et al. 2006]. We present a new explicit crack-tracking algorithm that is computationally cheaper than volumetric methods, utilizing local triangle mesh operations (see Figure 8). Our approach uses a surface tracking library [Da et al. 2014], but we have made significant modifications for crack tracking.

6.1 Mesh-Based Surface Tracking

The explicit surface tracking approach has become increasingly popular recently, as it preserves more geometric details than implicit methods. Interesting applications include viscoelastic

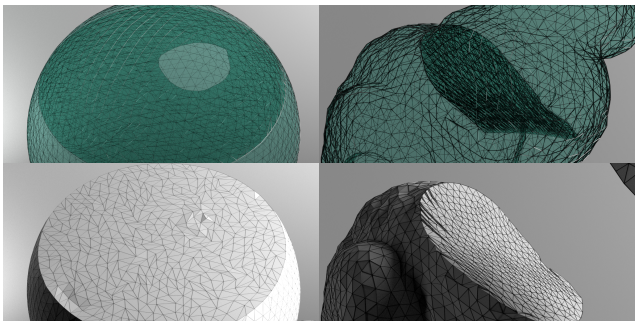


Figure 8: *Propagating fracture following a predefined path on sphere (**left**) and bunny (**right**). **Bottom** row shows the mesh quality of crack surface.*

objects [Wojtan et al. 2009], water [Brochu et al. 2010] and smoke [Brochu et al. 2012]. We extend this idea to rigid body brittle fracture simulation by explicitly tracking the crack propagation in the material space. We make the same assumption as Hegemann et al. [2013], namely that material failure starts from the boundary surface. Such an assumption ignores fracturing effects starting in the interior of the material, but is still acceptable for rigid body fracturing led by contact interaction or user cuts, which are the most common scenarios in computer graphics. It would be possible in future work to extend our method to support interior cracks by adding volumetric evaluation samples and inserting tiny ellipsoid meshes at crack starting positions.

6.2 Contact Force Model and Fracture Criteria

As we use a velocity level LCP-based rigid body solver to resolve rigid body contact. The physical outputs are contact impulses $\mathbf{I}_c$ instead of contact forces $\mathbf{f}_c$. To compute the corresponding contact forces, we estimate the contact duration t_d similarly to Koschier et al. [2014]. Then we assume a constant force acting during the whole contact period, which leads to a simple contact force model $\mathbf{f}_c = \mathbf{I}_c/t_d$. After we transform all contact forces for an object to its material space, we use them as traction boundary conditions for the elastostatic problem of that object and solve for $\boldsymbol{\rho}$ of the layer potential as described in § 4.3.

To open and propagate a crack, we evaluated two different cracking criteria, the Rankine condition and Griffith energy minimization. A shape optimization-based minimal energy approach gives rise to a direct way for crack propagation under the gradient descent flow, but suffers from inevitable volume loss depending on surface resolution, which is also evident in Hegemann et al.'s results [2013]'s results. By making use of the Rankine condition, we can control the volume loss, which is independent of surface resolution, by a user-defined crack thickness. Such parameter can be set very small to meet the visualization requirement.

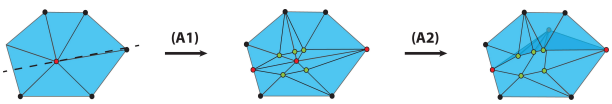


Figure 9: ***crack initiation** first applies local remeshing (**A1**) and then propagates active crack node into the material (**A2**).*

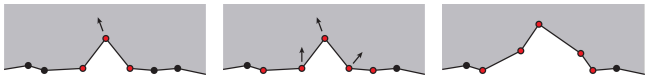


Figure 10: ***crack extension** extends the crack line when the end point either satisfies Rankine condition or a smoothness criterion.*

Algorithm 2 *Crack_Initiation*($\mathcal{N}$)

```
clear fracture line list L
sort N into descending order based on principal stress
for each node v in N
    if v lies in 1-ring of lines in L
        continue
    else
        l ← Open_Crack(v)
        add l into L
    end if
end for
return L
```

6.3 Fracture Initiation and Propagation

After solving the layer potential problem, we apply stress analysis to each vertex on the boundary surface. We choose vertices whose principal stress value is larger than a user-specified threshold, and maximal among all its 1-ring neighbours, as candidates to open cracks. Once we have a candidate list, we process them sequentially. To open a crack at a given vertex, we first compute the cutting line on its 1-ring triangle by the crack plane. If the cutting line is almost aligned with a 1-ring edge, we will replace it with the edge in order to avoid a degenerate triangle. Next, we offset the crack plane along its normal and opposite direction by a thickness parameter specified by the user. We then compute how these two planes intersect the vertex's 1-ring edge and apply local remeshing, as illustrated in Figure 9: the three red coloring nodes will compose a new crack line, from which the center node will be an active node which can evolve into the material while the other two will be inactive at this moment to represent the ends of the crack line. Once a crack line is generated, it will be stored in a crack line list. An overview of our fracture initiation procedure can be found in Algorithm 2.

To evolve a crack line in the material space, we propose a crack velocity model as follows and use it to propagate each active node on the crack line.

$$\mathbf{v}_{cp} = \alpha(\overline{\mathbf{h} - proj_{\mathbf{e}}(\mathbf{h})}) \tag{9}$$

where $\mathbf{h}$ is the unit vector that bisects the angle composed of the corresponding crack node and its two nearby crack nodes, $proj_{\mathbf{e}}(\cdot)$ is the orthogonal projection onto the principal eigenvector $\mathbf{e}$ of stress tensor. The bisection strategy contributes to a relative smooth crack front, as there is no physical rule on how a crack node (a synthetic construct of our geometric representation) should propagate within the plane defined by the principal eigenvector of stress. α is a scalar product of principal eigenvalue, a control parameter regarding the mesh resolution and a scaling factor tuned by the user which we set to 0.5. We also check the direction of the propagation velocity vector to enforce movement into the material. The overline symbol denotes unit vector. As presented in Figure 10, the end points of a crack line can become active according to the following criteria: (a) Rankine condition; (b) crack line smoothness. If one end $\mathbf{e}$ of the crack line becomes active, we will apply a crack extension operation. We compute the cutting line on the 1-ring triangles of $\mathbf{e}$ and choose the one which spans a larger angle with the crack line to avoid an inverting crack. Then we apply the same crack plane offset cutting and remeshing operation as described in crack opening operation. Finally, we activate $\mathbf{e}$ and use the extended point as the new end of the crack line.

As presented in Algorithm 3, a substep time integration will be applied when we propagate the crack lines. During each substep, we propagate each crack line sequentially and handle crack line

Algorithm 3 *Crack_Propagation*($\mathcal{L}$, sub_timestep)

```
dt ← sub_timestep
for each line l in L
    Update_Node_Position(l, dt)
    for each active node v in l
        if v hits boundary surface
            label v as inactive
        end if
    end for
    Local_Remeshing(l)
    Crack_Line_Smoothing(l)
    Feature_Preserved_Smoothing(l)
    Crack_Line_Extension(l)
    if number of active nodes in l equal 0
        Update_Mesh_Topology(l)
        remove l from L
    end if
end for
```

splitting and merging operations. When a crack line is touching the boundary surface, we apply vertex snapping operation similar to [Da et al. 2014] and mark the corresponding vertices on the crack line as inactive. The propagation process of a crack will stop if all vertices lying on it are inactive. An intuitive 2D illustration is presented in Figure 11. A merge operation occurs when a crack line forms a loop during the process of propagation as in Figure 12. We model this loop using crack circle and apply the same operation to it as crack line in terms of propagation, smoothing and remeshing. If two different crack lines intersect with each other, we apply a passive strategy to deactivate both of them.

When a crack circle degenerates to a point or a non-manifold line, we regard this as material separation and apply a separation operation by a local mesh connectivity update (see Figure 8 **left**). We also check if any two nearby nodes on a crack line or a crack circle are labeled as inactive. If so, we apply a topology change operation to account for crack cutting through material. Given two inactive crack node, we first use an average crack plane to cut through the material boundary surface colored in blue as in Figure 13. The crack plane will create a path on the surface mesh by intersection. A local remeshing operation will be applied following the intersection path. Next, we open the path and delete the two triangles sharing the same edge defined by the two inactive crack nodes on the crack surface colored in green. Finally, we close up the mesh by connecting the hanging vertices with new triangles.

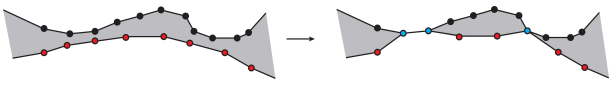


Figure 11: ***crack splitting*** *splits one crack line by deactivating crack nodes that touch the boundary surface.*

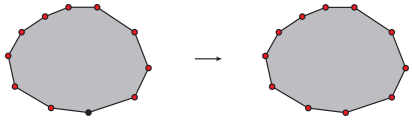


Figure 12: ***crack merging*** *turns a crack line into a crack circle when a loop is detected during the propagation.*

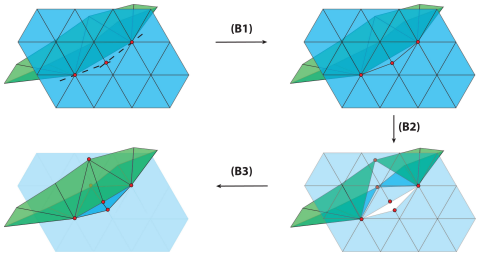


Figure 13: ***crack topology change*** *is applied to the non-manifold crack nodes. (**B1**) We use an interpolated cutting plane to locally remesh the boundary surface (blue). (**B2**) We then delete the triangles sharing the edge of two nearby nodes on the material surface (green) and thicken the inserted crack on the boundary surface. (**B3**) Finally, we stitch up the mesh to make it a closed manifold.*

6.4 Remeshing Operation

Mesh quality is always an essential component of physical simulation. During the substep time integration, a remeshing operation will be applied within each substep, as mentioned in Algorithm 3. Various metrics as well as remeshing approaches have been proposed for volumetric and surface meshes. Our remeshing operations are similar to what have been reported in [Brochu and Bridson 2009; Da et al. 2014], including edge splitting, edge collapsing, edge flipping. In addition, we make several extra modifications for valid operations to keep the crack line consistent before and after remeshing, which are listed as follow:

- During edge splitting, if one of the edge vertex is non-crack node, the new vertex is also non-crack node. Otherwise, the new vertex is a crack node.
- During edge collapsing, if both edge vertices are crack node or non-crack node, the merged vertex will be crack node or non-crack node respectively. Otherwise, the merged vertex will be a crack node.
- An edge flipping operation will take place if and only if at least one vertex of the candidate edge and one of the new edge are non-crack nodes.

Besides the above local remeshing operations, during each step we also apply a small amount of ridge preserving smoothing and crack line smoothing operations.

7 Results and Discussion

Both the single layer potential and Newton potential kernel functions in Equation 6 and their gradients are singular. In our implementation, we mollify the singular kernel by clamping the distance variable r to $10^{-5}m$ and apply Gaussian quadrature on each triangle when assembling the discretized linear system in Equation 8. To verify our physical model, we implement solvers for the 2D Laplacian problem in Matlab (Figure 4) and the 3D elastostatic problem in C++ (Table 1). The 2D tearing result, shown in the submission video, is composed of a plane stress model implemented in Matlab and a 2D ElTopo tracking library mexed from C++. The linear system is generated by a second kind Fredholm equation and is usually well-conditioned. We achieve convergence in less than 50 iterations (mostly 20 to 30) using a BiCG-STAB solver for all the examples we show in this paper. By replacing the matrix vector multiplication operations with FMM in each iteration, we achieve a linear running time with the model's surface resolution. In our FMM implementation, each octree node will be accompanied by two virtual spheres

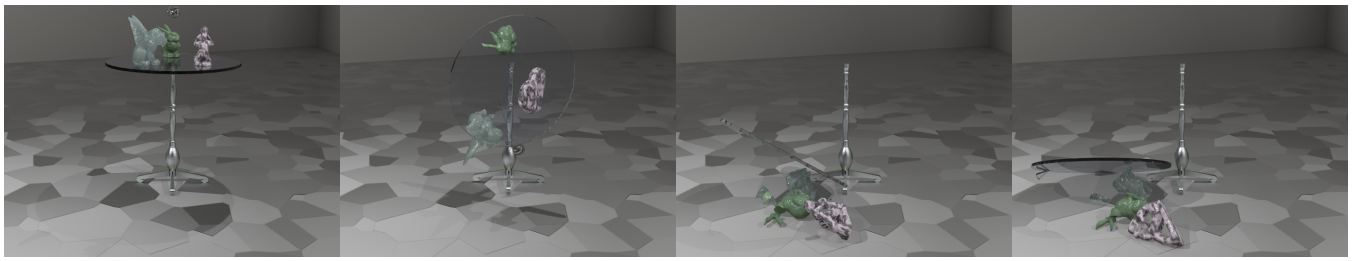


Figure 14: ***Three sculptures on a dinner table*** *are generated using our mesh based brittle fracturing method. Our explicit tracking approach preserves tiny but interesting surface details during fracture propagation.*

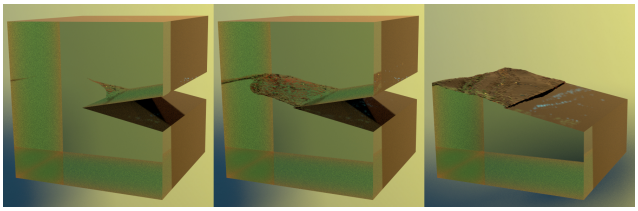


Figure 15: *Crack propagation inside the material due to constant external forces applied in the opposite direction at the top and bottom face.*

as illustrated in Figure 5. For a unit cube centered on the origin, we place both spheres, interior and exterior, co-centered with the cube. The size of the interior sphere and exterior sphere are set to 1.2 and 3.0 respectively. This stencil is applied to each octree node and scaled according to its size.

Our fracture propagation method is built on *LosTopos*, an explicit surface tracking library. As mentioned in § 6.3 and § 6.4, we modify the remeshing and topological changing strategies to adapt fracture propagation problem. Our current approach doesn't support adaptivity, so we pre-process all the downloaded mesh files at the beginning of each simulation (Figure 1, Figure 15, Figure 14, Figure 16), which is composed of several steps of remeshing and feature-preserved smoothing. In Figure 8, we show examples of crack propagation following a user defined path. The result is rendered in wireframe so the mesh updates can be clearly seen when the crack cuts through the model. During the physical simulation, we adopt the crack growth idea that brittle material requires significantly higher energy to start a new crack than to lengthen an existing crack by the same distance, as mentioned in [Smith et al. 2001], and use two different Rankine conditions for fracture initiation and fracture propagation. These two condition numbers as well as the crack thickness can be tuned by an artist, even varying them in material space, to achieve the desired fracturing pattern and simulation result.

We implemented a velocity level LCP based rigid body solver for

Case	LPSC(s)	BiCG-STAB	SEC(s)	CTC(s)
ceramic plate	176.349	$8 \sim 29$	3.383	0.877
mode I crack	404.492	22	17.008	1.472
glass goblet	363.396	$4 \sim 33$	8.411	1.92
hollow sphere	289.146	$4 \sim 26$	3.857	1.043
dinner table	397.443	$7 \sim 47$	14.736	1.185

Table 3: *Runtime cost details for the 3D fracture simulation cases included in this paper.* ***LPSC:*** *average layer potential solve cost;* ***BiCG-STAB:*** *BiCG-STAB iterations;* ***SEC:*** *average stress evaluation cost per fracture propagation step;* ***CTC:*** *average crack tracking cost per fracture propagation step.*

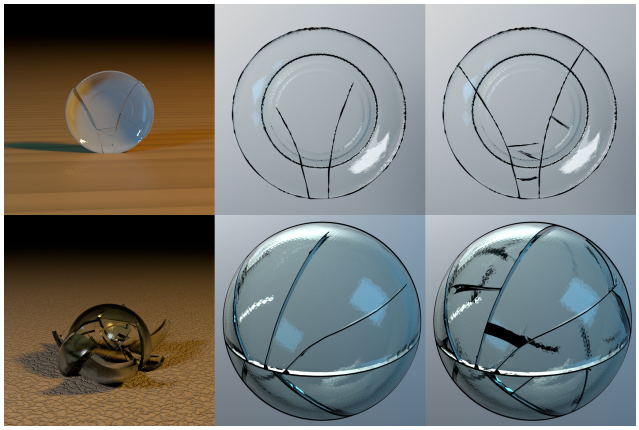


Figure 16: *Rigid body brittle fracture simulation of plate (**up**) and hollow sphere (**down**) with respective fracture propagation in the material space (**second & third column**).*

results shown in Figure 1, Figure 14 and Figure 16. In Figure 14, we apply different Lamé parameters and Rankine condition numbers to each object to achieve a different fracturing behavior. In Figure 1 and Figure 16, we demonstrate how the fractures propagate in the material space due to the rigid body contacts in the world space. We use the explicit Euler method for time integration and adopt a relatively small simulation timestep, 0.001s, in order to resolve enough contacts for fracture generation. Detailed runtime information of the simulation results is listed in Table 3. Our collision detection method is based on signed distance field versus point shell to handle general non-convex shapes. These modules can be replaced by other rigid body dynamic engines based on explicit mesh contact, like *Bullet*. In the example of Figure 15, we apply a pair of constant forces in the opposite direction at the top and bottom faces of the volumetric shape, and demonstrate how the crack initiates and propagates inside the material.

Limitations and Future Work Since our implementation is very aggressive in aiming to eliminate all volumetric degrees of freedom, there are some limitations in our proposed method. For example, we don't consider the case where material failure starts in the interior. Secondly, our passive strategy of dealing with crack merging can be further improved with more understanding of the physical behavior of this phenomenon. These limitations are items for future work. Additional themes for future investigation include developing more accurate quadrature rules for singular kernel integration, mesh adaptivity for explicit surface tracking, decoupling between the physical simulation's degrees of freedom and tracking mesh resolution, GPU parallization of kernel independent FMM, etc.

8 Conclusion

By combining an indirect boundary integral formulation, explicit surface tracking and a kernel-independent fast multipole method, we present the first effective method for rigid body brittle fracture using the boundary surface mesh only, and demonstrate its merits. Our method provides a novel direction for qualitative fracture simulation. It is accurate, and at the same time computationally economical, and it successfully resolves crack evolution in various settings. We have demonstrated the method's potential, and have pointed out a few directions for further exploration and improvement.

Acknowledgements

We thank the anonymous reviewers for their useful suggestions, and the Natural Sciences and Engineering Research Council of Canada for supporting this research.

References

ALPERT, B., BEYLKIN, G., COIFMAN, R., AND ROKHLIN, V. 1993. Wavelet-like bases for the fast solutions of second-kind integral equations. *SIAM J. Sci. Comput. 14*, 1 (Jan.), 159–184.

ANG, W.-T. 2014. *Hypersingular integral equations in fracture analysis*. Elsevier.

BAO, Z., HONG, J.-M., TERAN, J., AND FEDKIW, R. 2007. Fracturing rigid materials. *IEEE Trans. Vis. Comp. Graph. 13*, 2, 370–378.

BARNES, J., AND HUT, P. 1986. A hierarchical $o(nlogn)$ force-calculation algorithm. *Nature 324*, 446–449.

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*.

BLANDFORD, G. E., INGRAFFEA, A. R., AND LIGGETT, J. A. 1981. Two-dimensional stress intensity factor computations using the boundary element method. *International Journal for Numerical Methods in Engineering 17*, 3, 387–404.

BROCHU, T., AND BRIDSON, R. 2009. Robust topological operations for dynamic explicit surfaces. *SIAM J. Sci. Comput. 31*, 4, 2472–2493.

BROCHU, T., BATTY, C., AND BRIDSON, R. 2010. Matching fluid simulation elements to surface geometry and topology. *ACM Trans. Graph. (Proc. SIGGRAPH) 29*, 4, 1–9.

BROCHU, T., KEELER, T., AND BRIDSON, R. 2012. Linear-time smoke animation with vortex sheet meshes. In *Proc. ACM SIGGRAPH/Eurographics Symp. Comp. Anim.*, 87–95.

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

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

CHUNRUNGSIKUL, S. 2001. *Numerical quadrature methods for singular and nearly singular integrals*. PhD thesis, Brunel University.

DA, F., BATTY, C., AND GRINSPUN, E. 2014. Multimaterial mesh-based surface tracking. *ACM Trans. on Graphics (Proc. SIGGRAPH) 33*, 4, 112:1–11.

ERLEBEN, K. 2007. Velocity-based shock propagation for multibody dynamics animation. *ACM Trans. Graph. 26*, 2, 12.

FRANGI, A. 2002. Fracture propagation in 3d by the symmetric galerkin boundary element method. *International Journal of Fracture 116*, 4, 313–330.

GLONDU, L., MARCHAL, M., AND DUMONT, G. 2013. Real-time simulation of brittle fracture using modal analysis. *IEEE Trans. Vis. Comput. Graph. 19*, 2, 201–209.

GOLAS, A., NARAIN, R., SEWALL, J., KRAJCEVSKI, P., DUBEY, P., AND LIN, M. 2012. Large-scale fluid simulation using velocity-vorticity domain decomposition. *ACM Trans. Graph. 31*, 6, 148:1–9.

GREENGARD, L., AND ROKHLIN, V. 1987. A fast algorithm for particle simulations. *J. Comput. Phys. 73*, 2 (Dec.), 325–348.

HACKBUSCH, W., AND NOWAK, Z. 1989. On the fast matrix multiplication in the boundary element method by panel clustering. *Numerische Mathematik 54*, 4, 463–491.

HANRAHAN, P., SALZMAN, D., AND AUPPERLE, L. 1991. A rapid hierarchical radiosity algorithm. In *Proc. ACM SIGGRAPH*, 197–206.

HEGEMANN, J., JIANG, C., SCHROEDER, C., AND TERAN, J. M. 2013. A level set method for ductile fracture. In *Proc. ACM SIGGRAPH/Eurographics Symp. Comp. Anim.*, 193–201.

HOCKNEY, R. W., AND EASTWOOD, J. W. 1988. *Computer Simulation Using Particles*. Taylor & Francis, Inc., Bristol, PA, USA.

IRVING, G., TERAN, J., AND FEDKIW, R. 2004. Invertible finite elements for robust simulation of large deformation. In *Proc. ACM SIGGRAPH/Eurographics Symp. Comp. Anim.*, 131–140.

JAMES, D. L., AND PAI, D. K. 1999. ArtDefo: Accurate real time deformable objects. In *Proc. ACM SIGGRAPH*, 65–72.

JAMES, D. L., AND PAI, D. K. 2003. Multiresolution green's function methods for interactive simulation of large-scale elastostatic objects. *ACM Trans. Graph. 22*, 1.

JAMES, D. L., BARBIČ, J., AND PAI, D. K. 2006. Precomputed acoustic transfer: Output-sensitive, accurate sound generation for geometrically complex vibration sources. *ACM Trans. Graph. (Proc. SIGGRAPH) 25*, 3, 987–995.

KAUFMANN, P., MARTIN, S., BOTSCH, M., GRINSPUN, E., AND GROSS, M. 2009. Enrichment textures for detailed cutting of shells. *ACM Trans. Graph. (Proc. SIGGRAPH) 28*, 3.

KÖCKNER, A., BARNETT, A., GREENGARD, L., AND O'NEIL, M. 2013. Quadrature by expansion: A new method for the evaluation of layer potentials. *J. Comp. Phys. 252*, 332–349.

KEELER, T., AND BRIDSON, R. 2014. Ocean waves animation using boundary integral equations and explicit mesh tracking. In *Proc. ACM SIGGRAPH/Eurographics Symp. Comp. Anim.*, 9.

KIRKUP, S., AND YAZDANI, J. 2008. A gentle introduction to the Boundary Element Method in Matlab/Freemat. In *Proc. 10th WSEAS International Conference on Mathematical Methods, Computational Techniques and Intelligent Systems*, MAMECTIS'08, 46–52.

KOSCHIER, D., LIPPONER, S., AND BENDER, J. 2014. Adaptive tetrahedral meshes for brittle fracture simulation. In *Proc. ACM SIGGRAPH/Eurographics Symp. Comp. Anim.*

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.

MESSNER, M. 2008. *Time-dependent body forces within a boundary element formulation*. PhD thesis, Technische Universität Graz.

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

MÜLLER, M., AND GROSS, M. 2004. Interactive virtual materials. In *Graphics Interface 2004*, 239–246.

MÜLLER, M., MCMILLAN, L., DORSEY, J., AND JAGNOW, R. 2001. Real-time simulation of deformation and fracture of stiff materials. In *Proc. Eurographics 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. (Proc. SIGGRAPH) 32*, 4, 115:1–10.

NORTON, A., TURK, G., BACON, B., GERTH, J., AND SWEENEY, P. 1991. Animation of fracture by physical modeling. *The Visual Computer 7*, 4, 210–219.

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

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

PARK, S. I., AND KIM, M. J. 2005. Vortex fluid for gaseous phenomena. In *Proc. ACM SIGGRAPH/Eurographics Symp. Comp. Anim.*, 261–270.

PARKER, E. G., AND O'BRIEN, J. F. 2009. Real-time deformation and fracture in a game environment. In *Proc. ACM SIGGRAPH/Eurographics Symp. Comp. Anim.*, 156–166.

PAULY, M., KEISER, R., ADAMS, B., DUTRÉ, P., GROSS, M., AND GUIBAS, L. J. 2005. Meshless animation of fracturing solids. *ACM Trans. Graph. (Proc. SIGGRAPH) 24*, 3, 957–964.

PFAFF, T., NARAIN, R., DE JOYA, J. M., AND O'BRIEN, J. F. 2014. Adaptive tearing and cracking of thin sheets. *ACM Trans. Graph. (Proc. SIGGRAPH) 33*, 4, 110:1–9.

PORTELA, A., ALIABADI, M., AND ROOKE, D. 1992. The dual boundary element method: effective implementation for crack problems. *International Journal for Numerical Methods in Engineering 33*, 6, 1269–1287.

REZAYAT, M., SHIPPY, D., AND RIZZO, F. 1986. On time-harmonic elastic-wave analysis by the boundary element method for moderate to high frequencies. *Computer Methods in Applied Mechanics and Engineering 55*, 3, 349 – 367.

SIFAKIS, E., DER, K. G., AND FEDKIW, R. 2007. Arbitrary cutting of deformable tetrahedralized objects. In *Proc. ACM SIGGRAPH/Eurographics Symp. Comp. Anim.*, 73–80.

SLADEK, J., SLADEK, V., AND ATLURI, S. 2004. Meshless local petrov-galerkin method in anisotropic elasticity. *Comput Model Eng Sci 6*, 477–489.

SMITH, J., WITKIN, A., AND BARAFF, D. 2001. Fast and controllable simulation of the shattering of brittle objects. *Computer Graphics Forum 20*, 2 (June), 81–90.

STEINEMANN, D., OTADUY, M. A., AND GROSS, M. 2006. Fast arbitrary splitting of deforming objects. In *Proc. ACM SIGGRAPH/Eurographics Symp. Comp. Anim.*, 63–72.

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

SUN, T., THAMJAROENPORN, P., AND ZHENG, C. 2014. Fast multipole representation of diffusion curves and points. *ACM Trans. Graph. (Proc. SIGGRAPH 2014) 33*, 4.

TERZOPOULOS, D., AND FLEISCHER, K. 1988. Modeling inelastic deformation: viscolelasticity, plasticity, fracture. In *Proc. ACM SIGGRAPH*, vol. 22, 269–278.

VAN DER VORST, H. A. 1992. Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput. 13*, 2, 631–644.

WEISSMANN, S., AND PINKALL, U. 2010. Filament-based smoke with vortex shedding and variational reconnection. *ACM Trans. Graph. (Proc. SIGGRAPH) 29*, 3, 115:1–12.

WICKE, M., RITCHIE, D., KLINGNER, B. M., BURKE, S., SHEWCHUK, J. R., AND O'BRIEN, J. F. 2010. Dynamic local remeshing for elastoplastic simulation. *ACM Trans. Graph. (Proc. SIGGRAPH) 29*, 4, 49:1–11.

WOJTAN, C., THÜREY, N., GROSS, M., AND TURK, G. 2009. Deforming meshes that split and merge. *ACM Trans. Graph. (Proc. SIGGRAPH) 28*, 3, 76:1–76:10.

WU, J., WESTERMANN, R., AND DICK, C. 2014. Physically-based simulation of cuts in deformable bodies: A survey. In *Eurographics 2014 State-of-the-Art Report*, 1–19.

YING, L., BIROS, G., AND ZORIN, D. 2004. A kernel-independent adaptive fast multipole algorithm in two and three dimensions. *J. Comput. Phys. 196*, 2 (May), 591–626.

ZHENG, C., AND JAMES, D. L. 2010. Rigid-body fracture sound with precomputed soundbanks. *ACM Trans. Graph. (Proc. SIGGRAPH) 29*, 3.