Remeshing-Free Graph-based Finite Element Method for Fracture Simulation

AVIRUP MANDAL, Indian Institute of Technology Bombay, India
PARAG CHAUDHURI, Indian Institute of Technology Bombay, India
SUBHASIS CHAUDHURI, Indian Institute of Technology Bombay, India

Fig. 1. Our novel fracture simulation algorithm produces the intricate fracture patterns that result from the tearing of a loaf of bread.

Fracture produces new mesh fragments that introduce additional degrees of freedom in the system dynamics. Existing finite element method (FEM) based solutions suffer from increasing computational cost as the system matrix size increases. We solve this problem by presenting a graph-based FEM model for fracture simulation that is remeshing-free and easily scales to high-resolution meshes. Our algorithm models fracture on the graph induced in a volumetric mesh with tetrahedral elements. We relabel the edges of the graph using a computed damage variable to initialize and propagate fracture. We prove that non-linear, hyper-elastic strain energy density is expressible entirely in terms of the edge lengths of the induced graph. This allows us to reformulate the system dynamics for the relabeled graph without changing the size of the system dynamics matrix and thus prevents the computational cost from blowing up. The fractured surface has to be reconstructed explicitly only for visualization purposes. We simulate standard laboratory experiments from structural mechanics and compare the results with corresponding real-world experiments. We fracture objects made of a variety of brittle and ductile materials, and show that our technique offers stability and speed that is unmatched in current literature.

CCS Concepts: • **Computing methodologies** → **Physical simulation**.

1 INTRODUCTION

Fracture is a ubiquitous phenomenon in the real world. Everyday we come across several instances of fracture, ranging from the shattering of a brittle glass tumbler to the tearing of a soft loaf of bread. To recreate such real-world phenomena, realistic dynamic fracture simulations are used frequently in computer graphics. Simulating intricate fracture patterns and crack propagation often requires complex mathematical formulation. Incorporating both brittle and ductile fracture

Authors' addresses: Avirup Mandal, avirupmandal@ee.iitb.ac.in, Indian Institute of Technology Bombay, Main Gate Rd, IIT Area, Powai, Mumbai, Maharashtra, India, 400076; Parag Chaudhuri, paragc@cse.iitb.ac.in, Indian Institute of Technology Bombay, Main Gate Rd, IIT Area, Powai, Mumbai, Maharashtra, India, 400076; Subhasis Chaudhuri, sc@ee.iitb.ac.in, Indian Institute of Technology Bombay, Main Gate Rd, IIT Area, Powai, Mumbai, Maharashtra, India, 400076.

in a single framework needs even more sophisticated analysis. It has thus been an important research topic in both structural mechanics and computer graphics.

However, despite intensive research over the last two decades, the existing fracture techniques still suffer from scalability issues, poor stability and high computational cost. E.g., as the number of cracks grows in an object, the Finite Element Method (FEM) based approaches proposed in [O'Brien et al. 2002] and [O'Brien and Hodgins 1999] require remeshing to address the additional degrees of freedom (DOF). This poses challenges like the generation of thin (or sliver) elements, instability and heavy computational cost. To avoid these effects authors have explored remeshing-free techniques like eXtended Finite Element Method (XFEM) [Koschier et al. 2017] [Chitalu et al. 2020] and the Material Point Method (MPM) [Pauly et al. 2005] [Wang et al. 2019] [Hu et al. 2018a] [Wolper et al. 2019] [Wolper et al. 2020]. The system matrix of XFEM scales linearly with the introduction of new cracks, which in turn requires more computation time. Moreover, in XFEM large volume ratios in Heaviside enrichment leads to severe stability issues. On the other hand, the particle-based material point methods have shortcomings with respect to imposing boundary conditions, high computational cost and rigidity of fragmented segments. Contrary to the discrete formulations, peridynamics-based methods [Chen et al. 2018] [Levine et al. 2015] and boundary element methods (BEM) [Hahn and Wojtan 2015] [Hahn and Wojtan 2016] deal with solving the integral equations of continuum mechanics, thus bypassing the difficulty of imposing boundary conditions. But solving these singular integrals in peridymanics is extremely expensive in computation and therefore, is not suitable for low-resource applications. The use of BEM is limited to materials with large volumes simulating only brittle fracture.

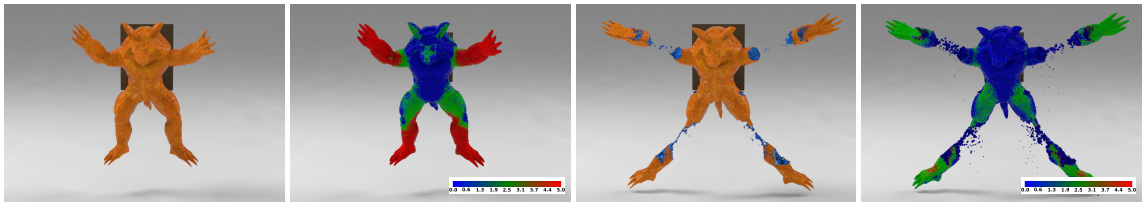
We perform our simulations on volumetric meshes with tetrahedral elements. This underlying mesh, on which all computation is performed, is called the computational mesh. This mesh induces a graph where its vertices become the nodes of the graph and the edges of the mesh become the edges of the graph. The computational mesh is never remeshed. This implies that, unlike XFEM, the number of degrees of freedom of the system does not change with the introduction of cracks, i.e., the size of the initial system matrix does not change. This allows our method to scale to high resolution models with very little extra computational overhead. The mesh that is rendered for visualization is the same as the computational mesh initially. This mesh, however, needs to be split and the fracture surfaces have to be reconstructed for rendering the fracture. This does not affect the computational mesh.

It has been shown in earlier work that the nodal forces of a discretized hyper-elastic FEM system can be completely represented in terms of a function of strain energy density along the edges of the induced graph [Reddy and Srinivasa 2015]. We use this fact and follow prior work in structural mechanics [Khodabakhshi et al. 2016] to define a purely edge-based damage variable, which describes the extent of the damage for any edge in the graph. We build over prior work to reformulate the strain energy density of damaged elements in 3D volumetric object meshes and then simulate the independent movement of the fractured pieces. Further in our model, we can generate and control the diffusion of cracks into the object by imposing a non-local fracture criterion.

We demonstrate the efficacy and robustness of our method by simulating realistic examples of both brittle and ductile fracture. Our method's ability to handle fracture of a variety of materials ranging from stiff glass to squishy jello makes it useful for a wide range of applications.

Our contributions are summarized as follows:

- We propose a novel remeshing-free, graph-based, computationally efficient and robust FEM solution for fracture simulation of 3D models.



(a) The model just prior to fracture. (b) Strain profile just prior to fracture. (c) Model post fracture. (d) Strain profile post fracture.

Fig. 2. Here we show frames from a simulation where we rip off the limbs of a jello armadillo. In the strain profiles, the red to blue colour gradient denotes the highest to lowest strain values. The model fractures where the strain is highest prior to fracture. Post fracture, the strain drops.

- We present a generalized version of a graph-based FEM algorithm that can simulate fracture of materials with both linear and non-linear strain energy density.
- We theoretically prove that hyper-elastic strain energy density can be represented in closed form as a function of the length of the edges of any object mesh.
- We extensively validate the correctness of our model against real-world structural mechanics experiments.
- We demonstrate via examples that our method serves as a unified FEM framework for simulating both brittle and ductile fracture.

The rest of the paper is organized as follows. We start by presenting a discussion of related works in Section 2. Subsequently, in Section 3 we first describe the theoretical formulation of graph-based FEM, and then detail the technical aspects of fracture via crack initiation and crack propagation. We discuss the implementation details about the visualization of fractured material, collision handling and the complete algorithm for our model in Section 4. Next, we present results for various kinds of materials undergoing fracture, generated using our method in Section 5. To validate our method and check its correctness, we compare our results with real-world benchmark fracture experiments and existing fracture simulation techniques in the literature. Finally, we conclude our paper by discussing the limitations of our work and future directions.

2 RELATED WORK

We first delve into existing methods for mesh-based fracture simulation in computer graphics. Next, we explore meshless particle-based fracture simulation methods. We conclude the section by reviewing the graph-based finite element analysis presented in material science literature.

2.1 Mesh-based Fracture Simulation

The inception of fracture simulation in computer graphics goes back to the seminal work on visco-elastic fracture by Terzopoulos and Fleischer [Terzopoulos and Fleischer 1988]. Early approaches propose to model brittle fracture by removing the springs in a mass-spring system [Aoki et al. 2004] [Norton et al. 1991] [Hirota et al. 2000] [Hirota et al. 1998] depending on a stress-based yield threshold. But in such a system, where point masses are connected by springs, the sudden removal of springs leads to significant visual artefacts. Moreover, visualization of the crack surface often requires the use of a tetrahedral marching algorithm and is therefore computationally expensive [Hirota et al.

2000] [Hirota et al. 1998]. Later, finite element method based solutions for simulating fracture have been more successful and different variations of those approaches are still widely used. The first breakthrough in FEM-based fracture came in the paper by O'Brien et al. [O'Brien and Hodgins 1999]. In their work, the authors present a nodal stress-based analysis for brittle fracture that was later extended further to ductile fracture [O'Brien et al. 2002] [Müller and Gross 2004]. Bao et al. [Bao et al. 2007] also present a method to simulate both brittle and ductile fracture. These FEM-based fracture techniques rely on splitting the elements of a tetrahedral mesh, which satisfy the fracture conditions and then remesh the entire fractured mesh with the crack opening. However, these approaches suffer from various geometric difficulties like remeshing near crack tips, generation of degenerate elements and forming ill-conditioned basis matrices. Subsequent FEM-based approaches employ different techniques to alleviate these problems, such as dynamic local mesh refinement to repair degenerate tetrahedra [Wicke et al. 2010], remeshing depending on gradient descent flow for finer fracture resolution [Chen et al. 2014], adaptive subdivision schemes for tetrahedral [Koschier et al. 2015] and triangular meshes [Pfaff et al. 2014] where remeshing is done around the high stress areas to improve fracture resolution.

FEM-based solutions that work with a virtual node algorithm (VNA) [Molino et al. 2004] duplicate elements and add extra degrees of freedom to facilitate partial or full crack openings, instead of splitting the mesh elements. Later, VNA-based methods are improved to incorporate cutting at lower than mesh resolution [Sifakis et al. 2007] and to robustly handle intersections passing through a node, edge or face through an adaptive element duplication approach [Wang et al. 2015]. Recently, Koschier et al. [Koschier et al. 2017] presented a remeshing-free extended finite element method (XFEM) based algorithm to simulate dynamic pre-defined cuts in a 3D mesh. Like VNA, XFEM too adds extra degrees of freedom using enrichment functions but the advantage of XFEM over VNA is while VNA duplicates the mesh element leading to erratic conservation of mass, XFEM splits the initial mass of an element into fragments. Chitalu et al. [Chitalu et al. 2020] expanded the pre-defined cuts to crack generation and propagation.

Contrary to processing fracture in volume elements like FEM, boundary element methods (BEM) simulate cracks on a surface mesh. Even though BEM found its way into graphics for simulating elastostatic deformable objects James et al. [James and Pai 1999], it has been used recently for fracture simulation. Hahn et al. [Hahn and Wojtan 2015] and Zhu et al. [Zhu et al. 2015] used BEM with stress intensity factors (SIFs) along crack fronts to successfully simulate brittle fracture. Later Hahn et al. [Hahn and Wojtan 2016] expanded their work to develop an algorithm for fast approximation of BEM brittle fracture, alleviating the high computational cost for solving singular integrals. Currently, to the best of our knowledge, there exists no work on BEM-based ductile fracture in graphics.

Peridynamics methods solve the integral equations of continuum mechanics. Unlike discrete differential-based methods e.g., FEM or MPM, which need to impose necessary boundary conditions for simulating fracture, it is trivial to model discontinuities using integral-based peridynamics. In initial work using these ideas, authors [Levine et al. 2015] approximate the behaviour of peridynamics using mass-spring systems to simulate brittle fracture. Later work [Chen et al. 2018] presents original peridynamics-based fracture algorithms to simulate brittle as well as ductile fracture.

2.2 Meshless Fracture Simulation

Among the meshless methods, material point methods have gained considerable recognition in recent years. The first major breakthrough of MPM in computer graphics came through the seminal work in snow simulation by Stomakhin et al. [Stomakhin et al. 2013]. Later MPM found its application in simulating a wide range of materials as such chocolate, lava, hybrid fluids and rubber to name a few. Interested readers can look at [Jiang et al. 2016] for a detailed overview of different MPM methods. Most recently, MPM has shown promising results in simulating fracture due to its ability to handle extreme topological change. Wu et al. [Hu et al. 2018a] propose the Compatible Particle-in-Cell (CPIC) algorithm,

which allows the simulation of sharp discontinuities inside a material and thus simulates dynamic material cutting. Later Wolper et al. [Wolper et al. 2019], introduce Continuum Damage Mechanics (CDM) with a variational energy-based formulation for crack evolution. This achieves a realistic dynamic fracture simulation for isotropic materials. Further study [Wolper et al. 2020] devoted to formulating non-local CDM to simulate anisotropic material fracture.

MPM methods reproduce realistic fracture phenomena but they are not well suited to rendering sharp boundaries. They not only have difficulty in enforcing essential boundary conditions, but they also suffer from the disappearance of intricate crack patterns due to excessive smoothing, high computational cost and rigidity of fragmented elements. Further, MPM methods struggle to simulate rigid object fracture.

2.3 Graph-based Finite Element Method

While simulating fractures in the material domain, classical FEM introduces degenerate elements due to remeshing and XFEM has several limitations like ambiguities of crack-tip enrichment in 3D, instabilities induced by large volume ratios in Heaviside enrichments, difficulties of handling branch enrichments in 3D. Moreover, in both of these methods, computational cost increases linearly with an increase in the number of cracks. While BEM and peridynamics overcome these problems by solving direct integrals, the use of BEM is limited to materials with large volumes simulating only brittle fracture and peridymanics has a high computational cost. Meshless methods have their limitations as described in the previous section.

Work in finite element analysis literature by Reddy and Srinivasa [Reddy and Srinivasa 2015] proposes a solution based on classical FEM to show that for any hyper-elasticity problem, the magnitude of the nodal forces can be written completely in terms of strain energy density along the edges composing the elements, while force directions are along the unit vectors corresponding to the edges. Using this idea in subsequent work, authors [Khodabakhshi et al. 2016] introduce Graph-based Finite Element Analysis (Gra-FEA) where a damage variable corresponding to the edges of composing elements is used. It is based on the strain threshold and thus can simulate the fracture of an object by weakening the material. The advantage of using Gra-FEA is that it requires no remeshing, while adding little computational overhead on FEM and it still retains all the advantages of FEM. The dependency of Gra-FEA on the underlying mesh structure is thoroughly studied in another work [Khodabakhshi et al. 2019]. Our work introduces Gra-FEA to the visual simulation of fracture for computer graphics. Existing literature referred to above on Gra-FEA only focuses on 2D materials consisting of triangular elements with linear strain energy density. We extend the idea to the 3D domain and our fracture algorithm can handle linear as well as non-linear strain energy density. The original Gra-FEA formulation is restricted to simulating fracture for brittle materials. Our algorithm is capable of simulating fracture for both brittle and ductile materials. We believe, to the best of our knowledge, that this is the first work in computer graphics to simulate the fracture of 3D brittle, as well as ductile objects, with linear or non-linear strain energy density using graph-based FEM.

3 GOVERNING METHODS

We derive the formulation of graph-based FEM from the theory of classical FEM, in brief. Then we delve into the details of fracture generation using graph-based FEM. In our work, we always use tetrahedral finite elements with a linear basis for domain discretization.

Symbol	Definition
$\mathbf{F} = \mathbf{I} + \nabla_{\xi}\mathbf{u}$	Deformation gradient
$J = \det(\mathbf{F})$	Relative volume change
$\mathbf{C} = \mathbf{F}^T\mathbf{F}$	Right Cauchy-Green deformation tensor
$I_C = \text{trace}(\mathbf{C})$	First Right Cauchy-Green invariant
$II_C = \mathbf{C} : \mathbf{C}$	Second Right Cauchy-Green invariant
$III_C = \det(\mathbf{C})$	Third Right Cauchy-Green invariant
$\mathbf{E} = \frac{1}{2}\left(\mathbf{F}^T\mathbf{F} - \mathbf{I}\right)$	Lagrangian finite strain tensor
$\mathbf{P}(\mathbf{F}) = \frac{\partial \Psi}{\partial \mathbf{F}}$	First Piola-Kirchhoff stress tensor

Table 1. Quantities frequently used in FEM

3.1 Classical FEM

Consider a three-dimensional domain $\Omega \in \mathbb{R}^3$ that is discretized into a mesh of n_{tet} tetrahedra. Moreover, let $\mathcal{N}$ be the set of nodes n_v shared by the tetrahedral mesh elements Δ_e. A displacement function, $\mathbf{u} : \Omega \times [0, \infty) \longrightarrow \mathbb{R}^3$, can be defined as a mapping from a material point $\xi \in \Omega$ to its deformed location $\mathbf{x} \in \Omega_t$ in the world space. Ω_t represents the world space at time t. At a certain timestep $t \in [0, \infty)$, the displacement function can be represented as

$$\mathbf{u}(\xi, t) = \sum_{i \in \mathcal{N}} \mathbf{N}_i(\xi)\mathbf{u}_i(t), \tag{1}$$

where $\mathbf{N}_i(\xi)$ is the shape function and $\mathbf{u}_i$ is the displacement vector at the node i. For the sake of brevity of notation, we drop the shape (ξ) and time (t) parameters for the shape and displacement functions in further discussion.

The core idea of deformable object simulation is that when an object mesh is deformed from its rest position, a hyper-elastic strain energy density Ψ, develops inside it. The strain energy density produces internal elastic force $\mathbf{f}^{int}$, which in turn tries to restore the rest shape of the mesh. Thus, the system dynamics of a deformed object can be written in Lagrange's form as

$$\mathbf{M}\ddot{\overline{\mathbf{u}}} + \mathbf{H}\dot{\overline{\mathbf{u}}} + \mathbf{f}^{int} = \mathbf{f}^{ext} \tag{2}$$

$$\overline{\mathbf{u}} = \left(\mathbf{u}_1^T \dots \mathbf{u}_{n_v}^T\right)^T \tag{3}$$

Here $\mathbf{M}$ and $\mathbf{H}$ are the mass and damping matrices of the full system. The external and internal force vectors are represented by $\mathbf{f}^{ext}$ and $\mathbf{f}^{int}$ respectively.

Interested readers can look into the review papers [Sifakis and Barbic 2012] [Kim and Eberle 2020] for further details. In this work, we developed our fracture simulation model based on works by Smith et al. [Smith et al. 2018], Sin et al. [Sin et al. 2013] and Bargteil et al. [Bargteil et al. 2007]. Table 1 summarizes some of the quantities that we use frequently in FEM simulations.

3.2 Graph-based FEM

Let the hyper-elastic strain energy density of a tetrahedral element, Δ_e, be Ψ^e and $\mathbf{f}_{e_i}^{int}$ be the internal elastic force acting on the i^{th} node of the element. Now it can be proved [Reddy and Srinivasa 2015] that nodal internal elastic forces of Δ_e can always be decomposed along the edge vectors connecting the nodes of Δ_e. The statement can be

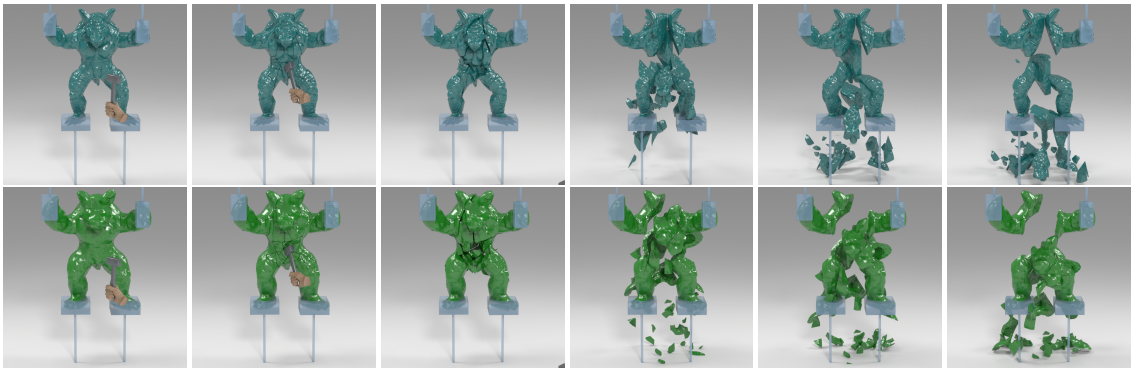


Fig. 3. Fracture of an armadillo model made of solid blue jade (top row) and porcelain shell (bottom row).

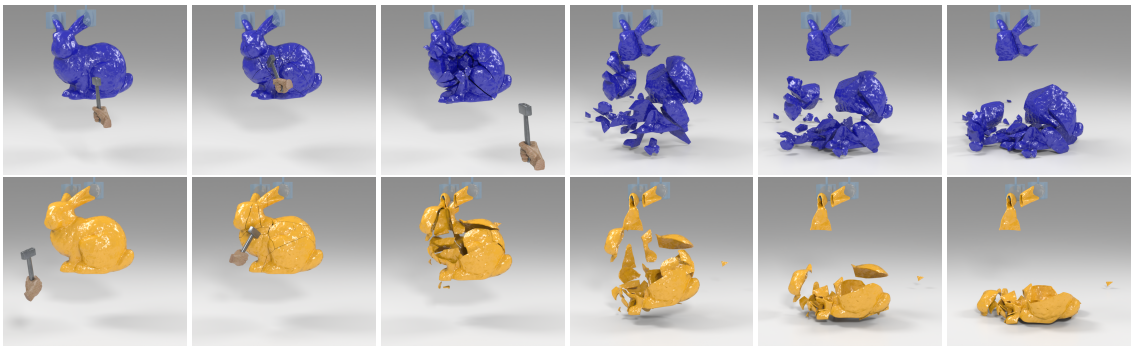


Fig. 4. Fracture of a bunny model made of solid glass (top row) and a porcelain shell (bottom row).

mathematically formulated as

$$\mathbf{f}_{e_i}^{int} = V_e \frac{\partial \Psi^e}{\partial \mathbf{r}_i^e} = 2V_e \sum_{\substack{j=1 \\ j \neq i}}^{n_e} \frac{\partial \Psi^e}{\partial \left(d_{ij}^e\right)^2} d_{ij}^e \hat{\mathbf{d}}_{ij}^e \tag{4}$$

where $\mathbf{r}_i^e$ is the world position vector for node i of element Δ_e. The parameter $d_{ij}^e = ||\mathbf{r}_i^e - \mathbf{r}_j^e||$ denotes the distance between the i^{th} and j^{th} nodes of Δ_e and $\hat{\mathbf{d}}_{ij}^e$ is unit vector along d_{ij}^e. Moreover, V_e and n_e represent the volume and the number of nodes of Δ_e respectively. In Equation 4 we can get to the rightmost term from the middle term by a few algebraic manipulations. We first take derivatives with respect to squared edge lengths and then apply a chain rule. Please check the supplemental document for a more detailed derivation. It can be seen from Equation 4 that the magnitude of the nodal internal elastic forces depends only on the derivative of the strain energy density of the element while the force directions are along its edges.

Figure 5 shows two tetrahedral elements, $\Delta_e ABDC$ and $\Delta_e BEDC$, which share a common face. These can be thought of as being a part of a larger object mesh. We assume that the overall mesh is deformed and each node in the mesh develops internal elastic force. Now according to Equation 4, internal elastic force on the node B for $\Delta_e ABDC$ can be written along the edges of $\Delta_e ABDC$ as $\mathbf{f}^B_{\Delta_e ABDC} = \mathbf{f}^{BC}_{\Delta_e ABDC} + \mathbf{f}^{BA}_{\Delta_e ABDC} + \mathbf{f}^{BD}_{\Delta_e ABDC}$ (shown as dotted blue lines). Similarly, internal elastic force on the node B for $\Delta_e BEDC$ can be written along the edges of $\Delta_e BEDC$ as $\mathbf{f}^B_{\Delta_e BEDC} = \mathbf{f}^{BC}_{\Delta_e BEDC} + \mathbf{f}^{BD}_{\Delta_e BEDC} + \mathbf{f}^{BE}_{\Delta_e BEDC}$

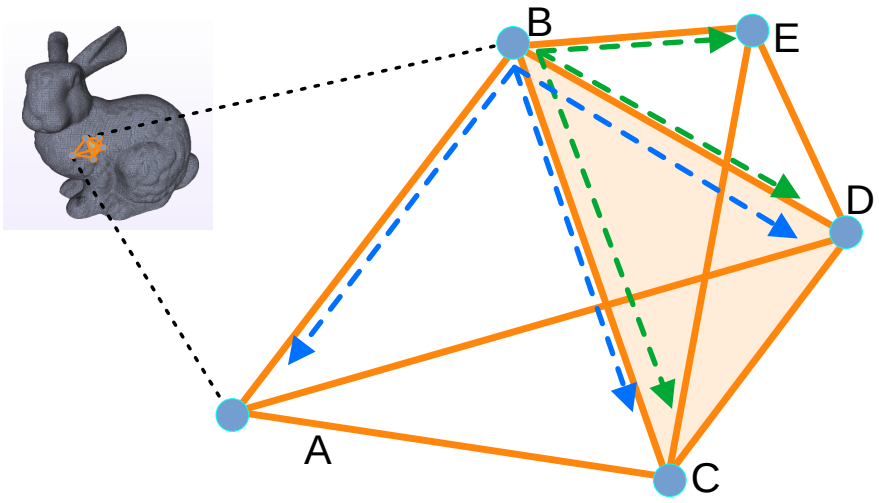


Fig. 5. Nodal force distribution along the edges for graph-based FEM.

(shown as dotted green lines). The distribution of nodal internal elastic forces for the entire mesh can be computed similarly.

However, for Equation 4 to hold, we need to show that hyper-elastic strain energy density Ψ^e can be completely represented using just d^e_{ij}. We prove this in the next section.

3.3 Proof for Edge Length Dependence of Strain Energy Density

Along with the Right Cauchy-Green deformation tensor $\mathbf{C}$ and the three invariants I_C, II_C and III_C, as defined in Table 1, let us introduce two more well-known anisotropic invariants [Weiss et al. 1996]

$$\begin{aligned} IV_C &= \mathbf{a}^T\mathbf{C}\mathbf{b} \\ V_C &= \mathbf{a}^T\mathbf{C}^T\mathbf{C}\mathbf{b} \end{aligned} \tag{5}$$

where $\mathbf{a}$, $\mathbf{b}$ are constant anisotropic fiber directions and they may or may not be equal to each other. Hyper-elastic energy density can generally be represented using a subset of these five invariants as $\Psi^e = f(I_C, II_C, III_C, IV_C, V_C)$. Therefore, in order to show that the hyper-elastic strain energy density can be expressed as a function of the length of edges of a mesh, it is sufficient to show that every element of the set of invariants can be expressed in closed form using only the length of the edges of the mesh in a graph-based FEM setting.

Theorem 3.1. *Every element of the set of invariants* $\mathbf{I}_V = \{I_C, II_C, III_C, IV_C, V_C\}$ *can be expressed in closed form using only the length of the edges of a mesh used in FEM.*

We present the complete proof of the theorem in the supplemental document.

3.4 Fracture with Graph-based FEM

In this section, we present our model for edge-based fracture derived from Graph-based FEM. We take advantage of the fact that the stress of an element is always a function of hyper-elastic strain energy density. We can find normal stress

in the direction of the edges of a tetrahedral element by using a suitable transformation. Next, depending on the state of the edge i.e., broken or unbroken, we manipulate the normal stress along the edges to degrade the strain energy density for a damaged element. Finally, using this updated strain energy density, we recalculate the elastic forces and tangent stiffness matrix for fracture simulation.

Using chain rule of differentiation and deformation gradient from Table 1, we can rewrite the Equation 4 as

$$\mathbf{f}_{e_i}^{int} = 2V_e \sum_{\substack{j=1 \\ j\neq i}}^{n_e} \sum_{k=1}^{3} \sum_{l=1}^{3} \left[\frac{\partial \boldsymbol{\Psi}^e}{\partial \left[\mathbf{F}\right]_{kl}} \frac{\partial \left[\mathbf{F}\right]_{kl}}{\partial \left(d_{ij}^e\right)^2} \right] d_{ij}^e \hat{\mathbf{d}}_{ij}^e \tag{6}$$

where $\frac{\partial \boldsymbol{\Psi}^e}{\partial \left[\mathbf{F}\right]_{kl}}$ are components of first Piola-Kirchhoff stress tensor of Δ_e (see Table 1). Thus, the edge-based elastic forces can be completely represented as a function of stress in Δ_e. This stress in turn depends on the derivative of element strain energy density, $\boldsymbol{\Psi}^e$ w.r.t. the deformation gradient $\mathbf{F}$. Therefore, by manipulating the stress components properly we can change the internal elastic forces and in turn simulate the fracture of an element.

Let the rectangular Cartesian components of the Piola-Kirchhoff stress tensor, $\boldsymbol{\sigma}_c^e$, be denoted as

$$\boldsymbol{\sigma}_c^e = \begin{bmatrix} \sigma_{xx}^e & \sigma_{yy}^e & \sigma_{zz}^e & \sigma_{xy}^e & \sigma_{xz}^e & \sigma_{yz}^e \end{bmatrix} \tag{7}$$

The set of normal stress along the edges is represented as

$$\boldsymbol{\sigma}_d^e = \begin{bmatrix} \sigma_{12}^e & \sigma_{13}^e & \sigma_{14}^e & \sigma_{23}^e & \sigma_{24}^e & \sigma_{34}^e \end{bmatrix} \tag{8}$$

where σ_{ij}^e represents the normal stress along the edge formed by nodes i and j, in Equation 8. Then the transformation of Cartesian stress to normal stress in the direction of an edge can be formulated as below [Reddy and Srinivasa 2015][Khodabakhshi et al. 2016]

$$\begin{aligned} \sigma_{ij}^e = {} & \sigma_{xx}^e \cos^2\phi_x + \sigma_{yy}^e \cos^2\phi_y + \sigma_{zz}^e \cos^2\phi_z \\ & + \sigma_{xy}^e \cos\phi_x \cos\phi_y + \sigma_{xz}^e \cos\phi_x \cos\phi_z + \sigma_{yz}^e \cos\phi_y \cos\phi_z \end{aligned} \tag{9}$$

Assuming that $\hat{\mathbf{d}}_{ij}^e$, $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$ denote unit vectors along the edge formed by nodes i and j of the tetrahedral element Δ_e, x-axis, y-axis and z-axis respectively, the following relations hold true

$$\cos\phi_x = \hat{\mathbf{d}}_{ij}^e \cdot \hat{\mathbf{x}}, \quad \cos\phi_y = \hat{\mathbf{d}}_{ij}^e \cdot \hat{\mathbf{y}}, \quad \cos\phi_z = \hat{\mathbf{d}}_{ij}^e \cdot \hat{\mathbf{z}} \tag{10}$$

Using Equation 9, we can write the final relation between two sets of stresses as

$$\begin{aligned} & \begin{bmatrix} \sigma_{12}^e & \sigma_{13}^e & \sigma_{14}^e & \sigma_{23}^e & \sigma_{24}^e & \sigma_{34}^e \end{bmatrix}^T \\ & = \mathbf{T} \begin{bmatrix} \sigma_{xx}^e & \sigma_{yy}^e & \sigma_{zz}^e & \sigma_{xy}^e & \sigma_{xz}^e & \sigma_{yz}^e \end{bmatrix}^T \end{aligned} \tag{11}$$

where, for convenience of notation, if we denote $\cos\phi$ as γ_ϕ, then

$$\mathbf{T} = \begin{bmatrix} \gamma^2_{\phi_x^{12}} & \gamma^2_{\phi_y^{12}} & \gamma^2_{\phi_z^{12}} & \gamma_{\phi_x^{12}}\gamma_{\phi_y^{12}} & \gamma_{\phi_x^{12}}\gamma_{\phi_z^{12}} & \gamma_{\phi_y^{12}}\gamma_{\phi_z^{12}} \\ \gamma^2_{\phi_x^{13}} & \gamma^2_{\phi_y^{13}} & \gamma^2_{\phi_z^{13}} & \gamma_{\phi_x^{13}}\gamma_{\phi_y^{13}} & \gamma_{\phi_x^{13}}\gamma_{\phi_z^{13}} & \gamma_{\phi_y^{13}}\gamma_{\phi_z^{13}} \\ \gamma^2_{\phi_x^{14}} & \gamma^2_{\phi_y^{14}} & \gamma^2_{\phi_z^{14}} & \gamma_{\phi_x^{14}}\gamma_{\phi_y^{14}} & \gamma_{\phi_x^{14}}\gamma_{\phi_z^{14}} & \gamma_{\phi_y^{14}}\gamma_{\phi_z^{14}} \\ \gamma^2_{\phi_x^{23}} & \gamma^2_{\phi_y^{23}} & \gamma^2_{\phi_z^{23}} & \gamma_{\phi_x^{23}}\gamma_{\phi_y^{23}} & \gamma_{\phi_x^{23}}\gamma_{\phi_z^{23}} & \gamma_{\phi_y^{23}}\gamma_{\phi_z^{23}} \\ \gamma^2_{\phi_x^{24}} & \gamma^2_{\phi_y^{24}} & \gamma^2_{\phi_z^{24}} & \gamma_{\phi_x^{24}}\gamma_{\phi_y^{24}} & \gamma_{\phi_x^{24}}\gamma_{\phi_z^{24}} & \gamma_{\phi_y^{24}}\gamma_{\phi_z^{24}} \\ \gamma^2_{\phi_x^{34}} & \gamma^2_{\phi_y^{34}} & \gamma^2_{\phi_z^{34}} & \gamma_{\phi_x^{34}}\gamma_{\phi_y^{34}} & \gamma_{\phi_x^{34}}\gamma_{\phi_z^{34}} & \gamma_{\phi_y^{34}}\gamma_{\phi_z^{34}} \end{bmatrix} \tag{12}$$

The matrix $\mathbf{T}$ is calculated at every timestep and it is invertible for non-degenerate tetrahedral elements, i.e., elements with non-zero volume.

The criteria for fracture in graph-based FEM is that if the weighted average of normal stress along the direction of any edge exceeds the critical stress threshold σ^e_{thres}, a crack is to be formed on that edge. First, we explain the method to calculate the weighted average of normal stress. Let $\mathbf{r}_{cen}$ be the position of the centroid of any tetrahedron of interest, Δ_{cen}, in the mesh at a particular time step t' during simulation. The weighted average of normal stress for Δ_{cen} can then be calculated as

$$\boldsymbol{\sigma}^e_{cen} = \sum_{||\mathbf{r}-\mathbf{r}_{cen}|| \leq R_d} \omega(\mathbf{r}-\mathbf{r}_{cen})\boldsymbol{\sigma}^e_c(\mathbf{r}) \tag{13}$$

where ω is a weight kernel centered at $\mathbf{r}_{cen}$ and R_d is support of the kernel. In our current implementation, we use a simple linear average kernel for ω for easier calculation. Other kernels e.g., a Gaussian kernel, can also be used. The exact diffusion fracture pattern (explained in the next paragraph) will be different for different kernels. Vector $\mathbf{r}$ and tensor $\boldsymbol{\sigma}^e_c(\mathbf{r})$ respectively refer to the position of centroid and PK1 stress tensor of any tetrahedron residing inside the kernel support R_d, at the same time step t'.

Using Equations 11 and 13 the fracture criteria can, therefore, be defined as [Khodabakhshi et al. 2019]

$$\sigma^{e^*}_{ij} = \left[\mathbf{T}\left[\boldsymbol{\sigma}^e_{cen}\right]^T\right]_{ij} \geq \sigma^e_{thres} \tag{14}$$

where $\sigma^{e^*}_{ij}$ is a component of the weighted average of normal stress along the edge formed by nodes i and j. As the support of kernel, R_d, increases, the cracks become more diffused i.e., they spread more inside the object [Khodabakhshi et al. 2019]. For $R_d = 0$, the fracture criterion depends only on the stress value of the tetrahedron of interest, Δ_{cen}, resembling a local fracture. The effect of kernel size R_d on fracture is shown in Figure 6. The top image shows an undamaged doormat model. In the middle and bottom images, the doormat has been torn apart by applying the same force at the free left end. As evident from the figure, in the middle image, where $R_d = 0$, the fracture is localized in a nearly horizontal tear at one place. On the contrary, in the image at the bottom, where $R_d = 5$, the cracks have been propagated over the entire body of the doormat, in a diffused pattern.

Now for each edge of the mesh, we maintain a binary damage variable, $\zeta \in \{0, 1\}$. A value of $\zeta = 0$ corresponds to an unbroken edge, while $\zeta = 1$ implies a crack is formed on the edge.

$$\zeta = \begin{cases} 0 & \text{if } \sigma^{e^*}_{ij} < \sigma_{thres} \\ 1 & \text{if } \sigma^{e^*}_{ij} \geq \sigma_{thres} \end{cases} \tag{15}$$

The number of broken edges in a tetrahedral element may vary from zero to six. Once an edge is broken, it will not get repaired in subsequent iterations and all the tetrahedra that share the broken edge will be considered damaged. As

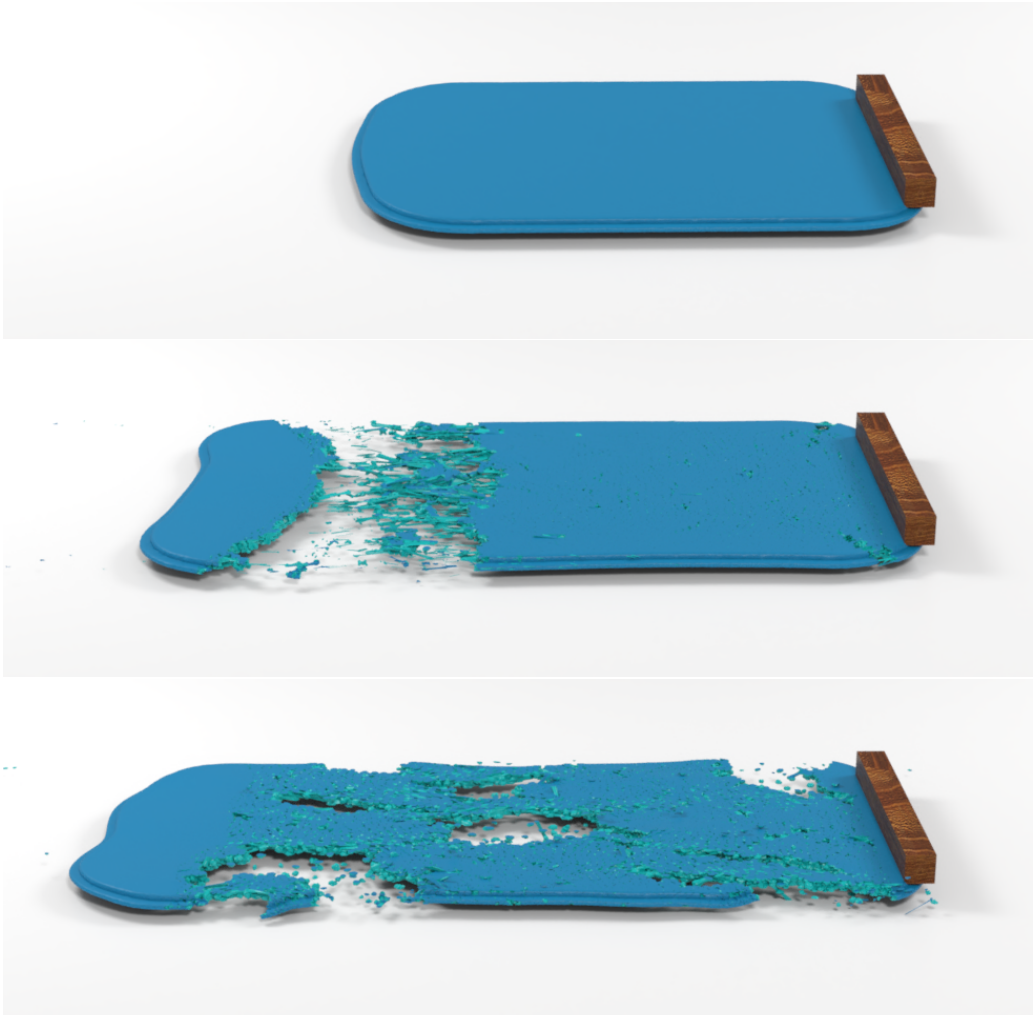


Fig. 6. Effect of kernel size R_d on fracture. We show the tearing of a PVC doormat (top) with different sizes of kernel support ($R_d = 0$ in middle, $R_d = 5$ at the bottom). As the size of kernel support increases, the cracks become more diffused over the object mesh.

an example, a shared edge fracture is depicted in Figure 7, using a simplified, two-dimensional example. Two edges in the mesh, coloured red, are broken. These broken edges are shared by triangles 1 (two edges), 2 (one edge) and 3 (one edge). In the subsequent iterations, all three triangular elements will be considered damaged with varying degrees of damage. Whenever an edge, e_{br}, gets broken, the corresponding normal stress of that edge becomes zero for all elements sharing that edge. Using Equations 11 and 15, we can write

$$\begin{aligned} &\begin{bmatrix} \sigma^e_{12_{frac}} & \sigma^e_{13_{frac}} & \sigma^e_{14_{frac}} & \sigma^e_{23_{frac}} & \sigma^e_{24_{frac}} & \sigma^e_{34_{frac}} \end{bmatrix}^T \\ &= \text{diag}\begin{bmatrix} \zeta_{12} & \zeta_{13} & \zeta_{14} & \zeta_{23} & \zeta_{24} & \zeta_{34} \end{bmatrix} \\ &\begin{bmatrix} \sigma^e_{12} & \sigma^e_{13} & \sigma^e_{14} & \sigma^e_{23} & \sigma^e_{24} & \sigma^e_{34} \end{bmatrix}^T \quad \forall e_{br} \in \Delta_e \end{aligned} \tag{16}$$

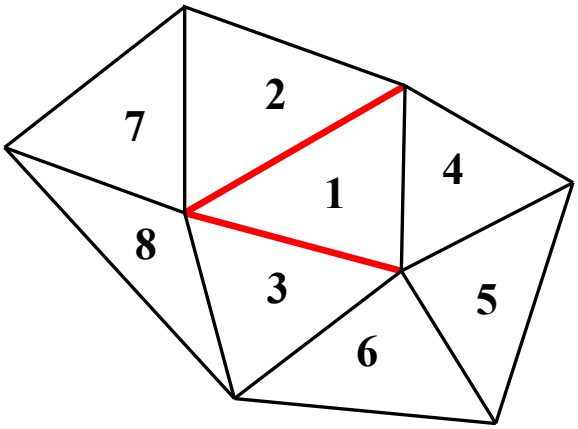


Fig. 7. Fractured edges shared multiple elements.

where ζ_{ij} and $\sigma^e_{ij_{frac}}$ are damage variable and stress after fracture along the edge formed by nodes i and j. In matrix form Equation 16 is written as

$$\left[\boldsymbol{\sigma}^e_{d_{frac}}\right]^T = \zeta \left[\boldsymbol{\sigma}^e_d\right]^T = \zeta \mathbf{T} \left[\boldsymbol{\sigma}^e_c\right]^T \tag{17}$$

where $\mathbf{T}$ is transformation matrix from Equation 12. Projecting back to Cartesian space, rectangular Cartesian components, $\boldsymbol{\sigma}^e_{c_{frac}}$ of the damaged Piola-Kirchhoff stress tensor can be written as

$$\left[\boldsymbol{\sigma}^e_{c_{frac}}\right]^T = \mathbf{T}^{-1} \left[\boldsymbol{\sigma}^e_{d_{frac}}\right]^T = \mathbf{T}^{-1} \zeta \mathbf{T} \left[\boldsymbol{\sigma}^e_c\right]^T \tag{18}$$

Graph-based FEM may seem similar to mass-spring systems and peridynamics, but it has a few key differences as explained in the supplemental document.

3.5 Fractured Strain Energy Density: Linear Elasticity

In graph-based FEM, instead of remeshing the initial mesh, we update the system equation of fractured mesh using a reformulation of elastic energy density to simulate fracture. In this section, we present the detailed derivation of hyper-elastic strain energy density for graph-based FEM in the case of linear elasticity problems. First, we derive it for undamaged condition. The expression for the damaged condition is a simple extension of that.

3.5.1 Undamaged Condition. In linear elasticity, the strain energy density for a tetrahedral element, Δ_e, the nodal internal force vector $\mathbf{f}^{int}_e$ of volume V_e can be written in terms of edge lengths as

$$\mathbf{f}^{int}_e = V_e \boldsymbol{\sigma}^e_c \mathbf{A}_2^T \mathbf{A}_1^T = V_e \boldsymbol{\sigma}^e_c \mathbf{B} = V_e \boldsymbol{\epsilon}^e_c \mathbf{E} \mathbf{B} \tag{19}$$

where $\mathbf{A}_1 = \left[\frac{\partial \mathbf{l}^e_d}{\partial \mathbf{u}_e}\right]^T$, $\mathbf{A}_2 = \left[\frac{\partial \boldsymbol{\epsilon}^e_c}{\partial \mathbf{l}^e_d}\right]^T$ and $\mathbf{B}^T = \mathbf{A}_1 \mathbf{A}_2$. Linear stress, $\boldsymbol{\sigma}^e_c$ and linear Cartesian strain, $\boldsymbol{\epsilon}^e_c$ are given by $\boldsymbol{\sigma}^e_c = \boldsymbol{\epsilon}^e_c \mathbf{E}$ and $\boldsymbol{\epsilon}^e_c = 1/2(\mathbf{F} + \mathbf{F}^T) - \mathbf{I}$. The parameter $\mathbf{l}^e_d$ denotes the edge length vector.

Similarly, we can formulate the tangent stiffness matrix in the graph-based FEM paradigm as below.

$$\mathbf{k}_e = \frac{\partial \mathbf{f}^{int}_e}{\partial \mathbf{u}_e} = V_e \mathbf{A}_1 \mathbf{A}_2 \mathbf{E} \mathbf{A}_2^T \mathbf{A}_1^T = V_e \mathbf{B}^T \mathbf{E} \mathbf{B} \tag{20}$$

More detailed derivations are given in the supplemental document.

3.5.2 Damaged Condition. Following the same line of arguments as presented in Equation 19 and Equation 20, the internal force and tangent stiffness matrix for fractured elements can be represented as

$$\mathbf{f}_{e_{frac}}^{int} = V_e \boldsymbol{\epsilon}_c^e \mathbf{T}^T \boldsymbol{\zeta} \mathbf{T}^{-T} \mathbf{E} \mathbf{T}^{-1} \boldsymbol{\zeta} \mathbf{T} \mathbf{B} \tag{21}$$

$$\mathbf{k}_{e_{frac}} = V_e \mathbf{B}^T \mathbf{T}^T \boldsymbol{\zeta} \mathbf{T}^{-T} \mathbf{E} \mathbf{T}^{-1} \boldsymbol{\zeta} \mathbf{T} \mathbf{B} \tag{22}$$

More details are available in the supplemental document.

3.6 Fractured Strain Energy Density: Non-linear Elasticity

Non-linear elasticity demands a different reformulation of the elastic energy density. In this section, we present two different approaches for this reformulation that work in the case of non-linear elasticity. There are also some approaches we tried that did not produce the correct result. These are instructive for understanding and are presented in the supplemental document for completeness.

3.6.1 First Approach — Linearization. Our first approach is based on the linearization of non-linear strain energy density. We reparameterize the material parameters μ and λ to reproduce the stress tensor, $\boldsymbol{\sigma}_c^e$, of linear elasticity according to Hooke's law.

$$\boldsymbol{\sigma}_c^e = 2\mu_{\text{Lamé}} \boldsymbol{\epsilon}_c^e + \lambda_{\text{Lamé}} \text{trace}\left(\boldsymbol{\epsilon}_c^e\right) \mathbf{I} \tag{23}$$

where coefficients $\mu_{\text{Lamé}}$ and $\lambda_{\text{Lamé}}$ are Lamé parameters in linear elasticity. The parameter $\boldsymbol{\epsilon}_c^e = 1/2(\mathbf{F} + \mathbf{F}^T) - \mathbf{I}$ is linear strain. Given Young modulus ($\mathbf{Y}$) and Poisson's ratio (ν), Lamé parameters can be defined as

$$\begin{aligned} \mu_{\text{Lamé}} &= \frac{\mathbf{Y}}{2(1+\nu)} \\ \lambda_{\text{Lamé}} &= \frac{\mathbf{Y}\nu}{(1+\nu)(1-2\nu)} \end{aligned} \tag{24}$$

For Neo-Hookean energy density [Smith et al. 2018] in Equation 25, if we set $\mu = 4/3\mu_{\text{Lamé}}$ and $\lambda = \lambda_{\text{Lamé}} + 5/6\mu_{\text{Lamé}}$, it becomes consistent [Smith et al. 2018] with Equation 23. With these linearized stress and strain values, we can rewrite the linearized Neo-Hookean energy density expression and follow the same line of arguments as presented earlier for fracture simulation.

$$\boldsymbol{\Psi}_{neo} = \frac{\mu}{2}(I_C - 3) + \frac{\lambda}{2}(J - \alpha)^2 - \frac{\mu}{2}\log(I_C + 1) \tag{25}$$

where $\alpha = 1 + \frac{\mu}{\lambda} - \frac{\mu}{4\lambda}$.

Similarly for Saint–Venant Kirchhoff energy density given in Equation 26, linearization [Kikuuwe et al. 2009] can be done by putting $\mu = \mu_{\text{Lamé}}$ and $\lambda = \lambda_{\text{Lamé}}$. We have simulated fracture using this method on Saint–Venant Kirchhoff energy density in Figure 8.

$$\boldsymbol{\Psi}_{stvk} = \frac{\lambda}{8}(I_C - 3)^2 + \frac{\mu}{4}(II_C - 2I_C + 3) \tag{26}$$

The derivations for the linearization of these strain energies are given in the supplemental document.

The main advantage of using this formulation is that even if an element gets fractured into two or more disjoint fragments, the residual force continues working on the remaining intact edges. However, for linearization operation, we assume infinitesimal linearized strain tensor $\epsilon = 1/2(\mathbf{F} + \mathbf{F}^T) - \mathbf{I}$ instead of Lagrangian finite strain tensor (see Table 1). As higher order terms of the strain tensor are ignored for linearization, it can not capture large deformations without

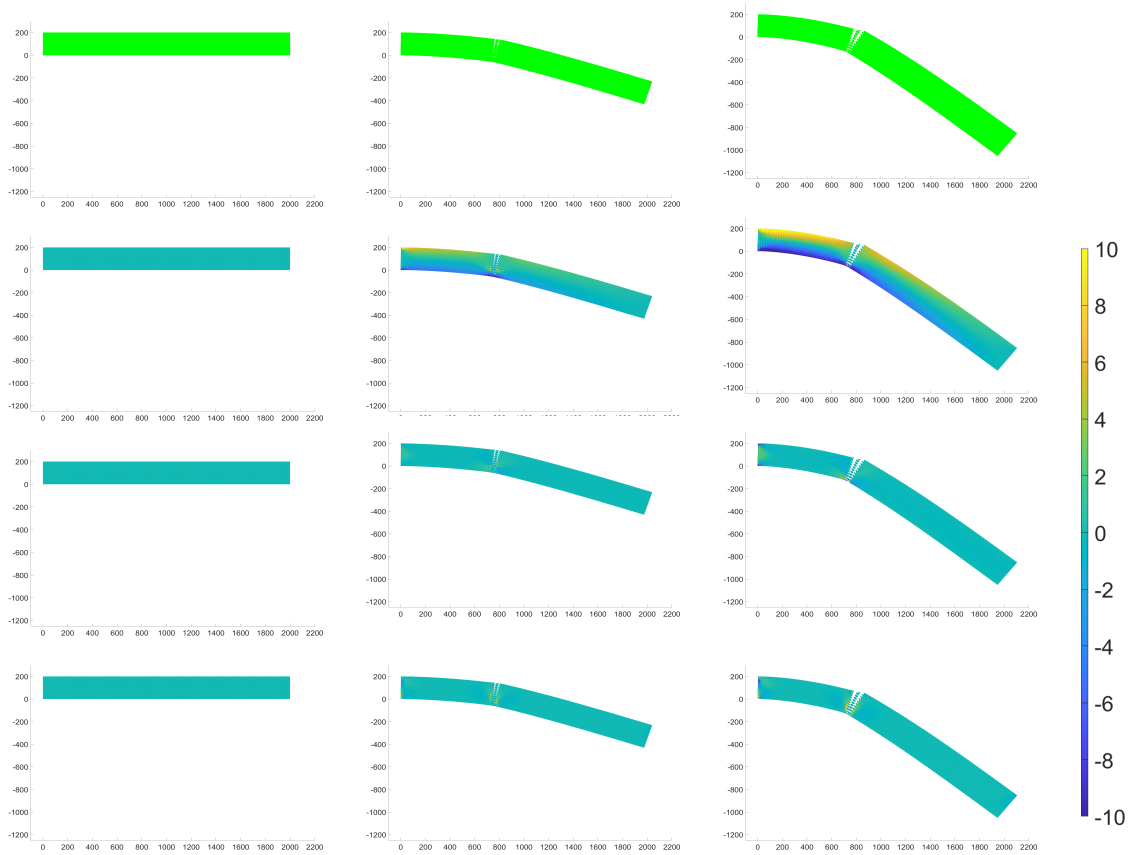


Fig. 8. Linearization of strain energy density for fracture simulation on a 2D bar (1^{st} row). The corresponding strain profiles are shown in $2^{nd}(\sigma_x)$, $3^{rd}(\sigma_{xy})$ and $4^{th}(\sigma_y)$ row respectively.

introducing significant artefacts. Thus, compared to non-linear models which use Lagrangian finite strain tensor, it produces more deformation artefacts.

3.6.2 Second Approach — Monotonic Degradation. Our second approach is based on the monotonic degradation of non-linear strain energy density. It is common in fracture simulation to degrade the energy density by a monotonic degradation function to allow material separation [Amor et al. 2009] [Miehe et al. 2015] [Wolper et al. 2019]. Thus, using Equation 8 and Equation 16 the hyper-elastic strain energy density for fractured elements can be defined as

$$\Psi^e_{frac} = \chi^e \left(\boldsymbol{\sigma}^e_{d_{frac}}, \boldsymbol{\sigma}^e_d \right) \Psi^e \tag{27}$$

The monotonic function $\chi^e\left(\boldsymbol{\sigma}^e_{d_{frac}}, \boldsymbol{\sigma}^e_d\right)$ signifies the fractured hyper-elastic strain energy density as a fraction of original one depending on the number of damaged edges.

$$\chi^e\left(\boldsymbol{\sigma}^e_{d_{frac}}, \boldsymbol{\sigma}^e_d\right) = \frac{\sum\limits_{\substack{m,n=1 \\ m<n}}^{4} |\sigma^e_{ij_{frac}}|}{\sum\limits_{\substack{m,n=1 \\ m<n}}^{4} |\sigma^e_{ij}|} \tag{28}$$

where i and j denote nodes of the tetrahedral mesh element, Δ_e. Further, this updated hyper-elastic strain energy density is used to calculate internal force and tangent stiffness matrix for fractured elements. We have simulated fracture using this method on Saint–Venant Kirchhoff energy density in Figure 9.

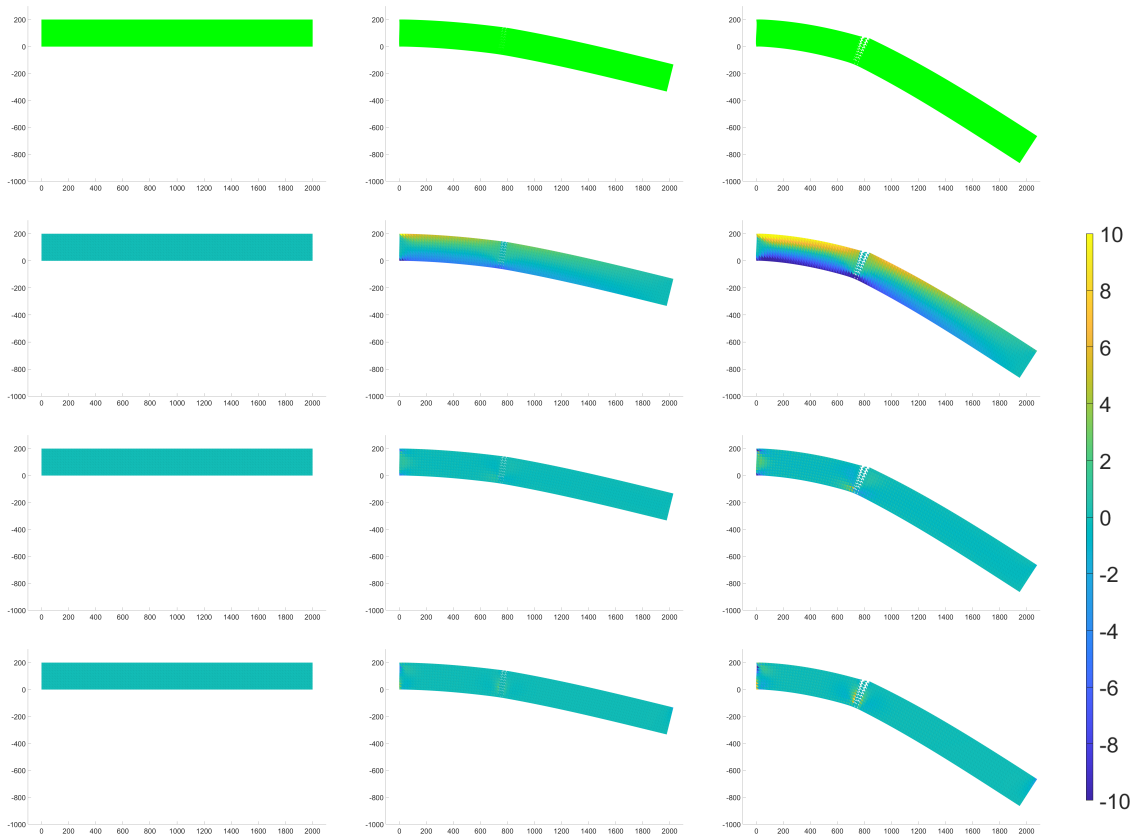


Fig. 9. Degradation of hyper-elastic strain energy density for fracture simulation on a 2D bar (1^{st} row). The corresponding strain profiles are shown in $2^{nd}(\sigma_x)$, $3^{rd}(\sigma_{xy})$ and $4^{th}(\sigma_y)$ row respectively.

The main advantage of this method is that we can use Lagrangian finite strain tensor for non-linear elasticity. Thus using this approach, large deformations can be better simulated without any artefacts contrary to the earlier linearization method. However, in this approach, we assign zero value to the fractured hyper-elastic strain energy

density i.e. $\Psi^e_{frac} = 0$, in Equation 27 when an element fractures into two or more disjoint fragments. Thus, in this case, when an element produces two or more disjoint segments, no residual force remains in the intact edges. This is a drawback compared to the previous linearization model where residual force continues working on the remaining intact edges even for disjoint fragments. However, any edge in an object mesh is shared by multiple elements all of which may not be fully damaged. The internal forces from these undamaged elements continue to work on those intact edges and thus, keep the nodes of the undamaged edges together for a fully split tetrahedron.

From Figure 8 and Figure 9, we can see that there is no significant change in the visual output of the fracture simulation regardless of the reformulation method used. Moreover, irrespective of methods used for updating strain energy density after fracture, both of these approaches use the rich discrete differential geometric approach [Srinivasa 2021] of graph-based FEM to simulate fracture. From the discussion of the two approaches, we can conclude that if our main objective is to simulate large deformation-based fracture, we should use the second approach. On the other hand, if our goal is to accurately capture the residual stress after fracture for some engineering applications, then the first approach should be preferred.

3.7 Plasticity

We follow the multiplicative plasticity model [Bargteil et al. 2007] for our work. In this model, the deformation gradient is split into an elastic and a plastic part

$$\mathbf{F} = \mathbf{F}_e\mathbf{F}_p \tag{29}$$

where forcing $\det(\mathbf{F}_p) = 1$, the volume of the element is preserved. Starting from an initial identity matrix, $\mathbf{F}_p$ is updated as follows

$$\mathbf{F}_p \longleftarrow \mathbf{F}_p \cdot \mathbf{V}\left(\det(\Sigma)^{-\frac{1}{3}}\,\Sigma\right)^{\beta}\mathbf{V}^T \tag{30}$$

where $\mathbf{U}\Sigma\mathbf{V}^T$ is SVD of $\mathbf{F}_e$. The exponent β is an function of current stress ($\boldsymbol{\sigma}^e$), yield stress (σ^e_{thres}), flow rate (ν) and hardening parameter (δ, K)

$$\beta = \text{clamp}\left\{\frac{\nu\left(||\boldsymbol{\sigma}^e||_2 - \sigma^e_{thres} - K\delta\right)}{||\boldsymbol{\sigma}^e||_2}, 0\ldots1\right\} \tag{31}$$

where ν and K are user-defined parameters. The term $K\delta$ determines the work hardening or softening. The term δ is initialized with value zero and incremented in each time step by $\delta \longleftarrow \delta + \Delta t||\boldsymbol{\sigma}^e||_2$.

4 IMPLEMENTATION

Next, We present details of how we create visualizations of the fracture simulation. Following that, we discuss collision detection for fractured pieces. Then we present a complete algorithm for our model of fracture simulation.

4.1 Surface Remeshing for Visualization

Our FEM computations are performed on the graph induced by the computational mesh, $\mathcal{M}_c$, that is never remeshed. However, to visualize the fracture we need to split the mesh used for visualization. For that purpose, a separate visualization surface mesh, $\mathcal{M}_v$, is maintained in addition to $\mathcal{M}_c$ and is the same as the outer surface of volumetric mesh initially. We clarify this distinction between the $\mathcal{M}_c$ and $\mathcal{M}_v$ meshes using a simple 2D example. Figure 10b depicts that when a fracture occurs, the system dynamics is evaluated on $\mathcal{M}_c$ with weakened elements. When we need to render the fractured elements, we split the $\mathcal{M}_v$ mesh that is used for visualization, as shown in Figure 10c and reconstruct

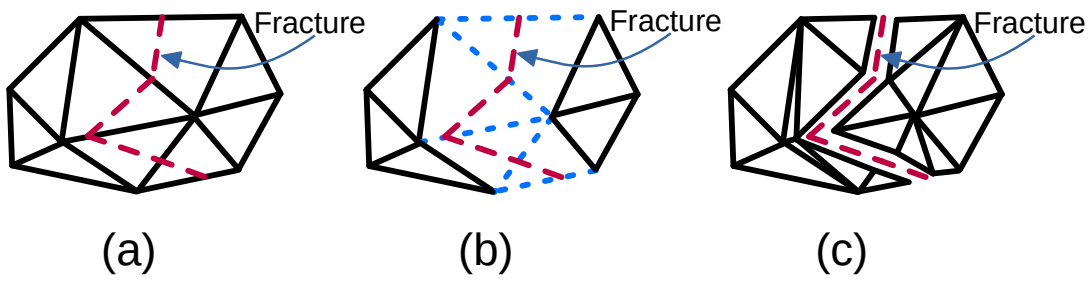


Fig. 10. (a) Original mesh with a fracture, (b) Computational mesh ($\mathcal{M}_c$), with damaged edges marked in blue, on which system dynamics get evaluated, (c) Mesh for visualization ($\mathcal{M}_v$) is split and the fracture surface is reconstructed.

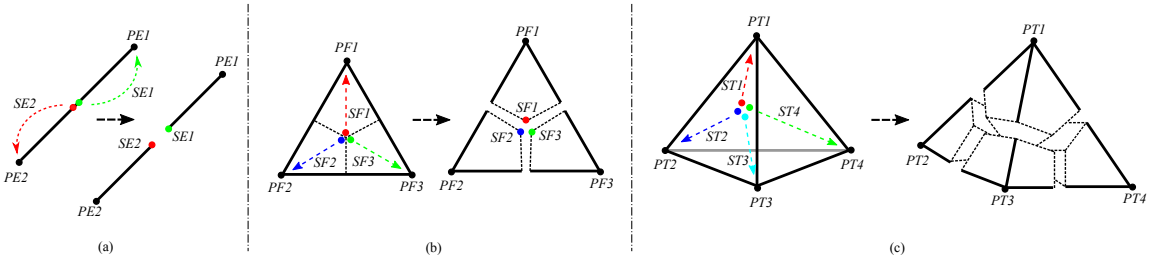


Fig. 11. Splitting of edge (left), face (middle) and tetrahedron (right) for visualization of fracture.

the fracture surface. Remeshing of $\mathcal{M}_v$ does not affect $\mathcal{M}_c$. Moreover, the reconstruction of the fracture surface for visualization adds very little computational overhead to the overall simulation cost. We have already explained how the system dynamics is computed on $\mathcal{M}_c$ in the presence of a fracture.

We now explain the details of how we split $\mathcal{M}_v$ for rendering. We use a node-associated four-split configuration of a tetrahedron, as shown in Figure 11c, for fracture visualization. When a tetrahedral element splits, the corresponding edges and faces of the element split too. We explore two different split strategies, as explained at the end of this section.

When normal stress along an edge, *PE1PE2* (Figure 11a), crosses the critical threshold, it splits into two segments *PE1SE1* and *PE2SE2*. *PE1* and *PE2* are original nodes in the mesh. Let us call them parent nodes. The other two newly generated nodes, *SE1* and *SE2*, are child nodes to *PE1* and *PE2*, respectively. The child nodes do not contribute to the system dynamics but rather follow the movements of their parent nodes. Similarly, when a face gets fractured into three segments (Figure 11b), three child nodes, *SF1*, *SF2* and *SF3* are assigned to their corresponding parent nodes, *PF1*, *PF2* and *PF3*. Similarly four child nodes, *ST1*, *ST2*, *ST3* and *ST4* (Figure 11c) are assigned to four parent nodes *PT1*, *PT2*, *PT3* and *PT4* for a damaged tetrahedron. It is important to note that a triangular face or tetrahedron may not split into all three or four segments during the simulation. For example, with reference to Figure 11b, if we assume that only both the segments containing node *PF1* get damaged i.e., edge *PE2PE3* remain undamaged, then in such a case, child node *SF1* follows the movement of parent node *PE1* but child node *SF2* and child node *SF3* follow the average movement of parent nodes *PE2* and *PE3*. When similar scenarios, although more complex, arise for a tetrahedron, we apply the same principle. All possible cases of different kinds of fractures are taken into account in our simulation.

In this work, we split edges, faces and tetrahedra with respect to their centroids. Furthermore, we also examined how the visualization output gets affected if an edge is split at a different position instead of in the middle. For this

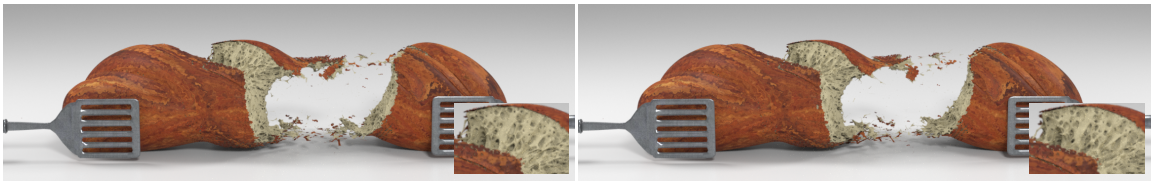


Fig. 12. The left column displays fracture where an edge is split in middle. The right column displays fracture where an edge is split in the ratio of the forces on its nodes. The fractured areas in each image are zoomed up in the inset for better visualization.

purpose, we split the edge such that the lengths of the sections are in inverse proportion to the ratio of its nodal forces. In Figure 12, in the left column, we render fracture on the bread model where all edges are split in the middle. In the right column, the same simulation is repeated with exact parameters and the edges are split in inverse proportion to the ratio of their corresponding nodal forces. As evident from the figure, there is no significant difference in the simulation outputs. We observed that both simulations require equivalent computation time.

4.2 Collision Detection

After the fracture, disjoint vertices have no internal elastic force acting on them. But external body forces e.g., gravity or impulse force, continue to be applied on all the vertices whether disjoint or not. So we need no special treatment for collision detection. Continuous Collision Detection (CCD) [Tang et al. 2010] [Tang et al. 2012] [Wang et al. 2020] is performed between $\mathcal{M}_v$ and the collider mesh (see Figure 3, 4). For fractured pieces colliding with the floor, we use Discrete Collision Detection (DCD) [Erleben et al. 2005] routine (see Figure 4). When the visualization mesh, $\mathcal{M}_v$, collides with another object mesh, we first calculate the impulse forces for each of them using the principles of CCD and DCD. Now, as shown in Figure 11c, each fractured segment of a tetrahedron has exactly one parent node. For each fractured segment, we accumulate all the impulse collision forces corresponding to that particular segment and apply the accumulated force to the parent node associated with the segment. However, as discussed in the previous section, for a partially fractured tetrahedron, there will be disjoint segments which contain more than one parent node. In that case, the accumulated force is equally distributed among all the parent nodes present in the fractured segments. Thus even though the visualization mesh, $\mathcal{M}_v$, accurately detects the collisions, the effects of the collisions are handled by the computational mesh, $\mathcal{M}_c$. Currently, we do not handle self-collision between the fractured pieces.

4.3 Algorithm

The full algorithm of our method is presented in Algorithm 1. We implement our model using the open-source Vega FEM library [Sin et al. 2013]. In the implementation of this algorithm, for solving the system dynamics we use an implicit backward Euler integrator with a conjugate gradient solver [Sin et al. 2013]. All the simulations are performed on an Intel Core i7-9750H CPU with 12 threads at 2.60 GHz. For visualization, the simulation results are raytraced in Houdini. Apart from the models obtained from the Stanford 3D scanning repository, the other 3D models used in this project are obtained from *free3d.com*, *sketchfab.com*, *turbosquid.com*, *cgtrader.com* and *lifesciencedb.jp*. Open-source software TetWild [Hu et al. 2018b] and TetGen [Si 2015] are used to generate the volumetric simulation meshes.

Algorithm 1: Remeshing-Free Graph-based Fracture

```
Initialize FEM simulation;
Let n_v be total no of vertices of M_c;
Let [x]_{n_v×1} be initial position vector;
while True do
    for Each element in M_c do
        Calculate stress along the edges σ^e_ij [as per Eq. 9];
        if σ^e_ij > σ_thres then
            Fracture the edge [as per Eq. 14];
            Update strain energy density Ψ^e [as explained in Sec. 3.5 & Sec. 3.6];
            Update internal force f^int_e;
            Update stiffness matrix k_e;
            Remesh the M_v for visualization [as explained in Sec. 4.1];
        end
        Resolve all collisions with M_v [as explained in Sec. 4.2] ;
        Calculate impulse force due to collision;
        Add all external forces to the vertices of M_c;
    end
    Build full system [M]_{n_v×n_v} [v]_{n_v×1} = [f]_{n_v×1};
    Solve for velocity vector [v]_{n_v×1};
    for Each vertex in M_c and M_v do
        Update position vector by [x]_{n_v×1} += Δt · [v]_{n_v×1};
    end
end
```

5 RESULTS

We present multiple visualizations of fracture simulation for a variety of materials to demonstrate the robustness of our method and the diversity it can handle. We present dynamic fracture simulation results which mimic real-world events like tearing a loaf of bread, collision of hard objects with solid jade models and objects made of porcelain shells. We also show the effect of applying tensile or impact forces to rubber-like and jello-like materials. Following that we discuss the effects of mesh resolution on both visualization and computational mesh during fracture simulation.

Next, we experimentally validate our model by recreating benchmark fracture experiments in simulation that are commonly performed in a structural mechanics laboratory, with our framework and compare the results with expected results from similar experiments performed in the real world. These experimental simulations are complete dynamics simulations that demonstrate the accuracy of our method. Finally, we compare our method with existing related works.

Mesh resolution and timing results for the simulations are reported in Table 2. The timestep for all our simulations is $5.0e-03$.

5.1 Dynamic Simulations

In a complete dynamic simulation, we take into account the effects of external forces and solve full system dynamics as presented in Equation 2 with mass & damping matrix regularizers. We use Neo-Hookean energy with monotonic degradation for ductile fracture simulation as it is better at preserving large deformation. StVK energy with linearization is used for brittle fracture simulation as brittle materials show very little deformation.

Simulation	***# Tetrahedra***	***sec/frame***	***ρ (kg/m^3)***	**Y *(Pa)***	***ν***	***R_d***	***σ_{th} (Pa)***	***σ_p (Pa)***
Loaf (Fig. 1)	620.6k	4.38	1.0e+03	1.0e+07	0.35	1.0	4.0e+06	1.0e+06
PVC doormat (Fig. 6)	234.4k	1.62	1.0e+03	1.0e+08	0.45	0.0 & 5.0	2.5e+06	1.5e+05
Jello armadillo (Fig. 2)	159.1k	1.09	1.0e+03	1.0e+06	0.3	2.0	1.5e+05	5e+03
Solid jade armadillo (Fig. 3)	4.7k	0.04	1.0e+03	1.0e+10	0.4	0.0	5.0e+07	-
Porcelain shell armadillo (Fig. 3)	41.1k	0.33	1.0e+03	1.0e+10	0.4	0.0	5.0e+07	-
Solid glass bunny (Fig. 4)	23.7k	0.18	1.0e+03	1.0e+10	0.45	0.0	5.0e+07	-
Porcelain shell bunny (Fig. 4)	38.8k	0.29	1.0e+03	1.0e+10	0.45	0.0	5.0e+07	-
Granite cylinder (Fig. 21)	45.8k	0.39	2.75e+03	4.5e+10	0.33	0.0	9.56e+06	-
A516 Gr.70 steel bar (Fig. 17, 19)								
Brittle at temperature $-40°C$	27.8k	0.24	7.7e+03	2.2e+11	0.28	0.0	6.55e+08	-
NVE 36 steel bar (Fig. 17, 19)								
Ductile at temperature $23°C$	27.8k	0.24	7.7e+03	1.9e+11	0.28	0.0	5.4e+08	8.3e+07

Table 2. Simulation parameter table. Parameters ρ, Y, ν and R_d (Equation 13) denote density, Young modulus, Poisson's ratio and support of the kernel for crack diffusion. Parameters σ_{th} and σ_p denote yield and plasticity threshold.

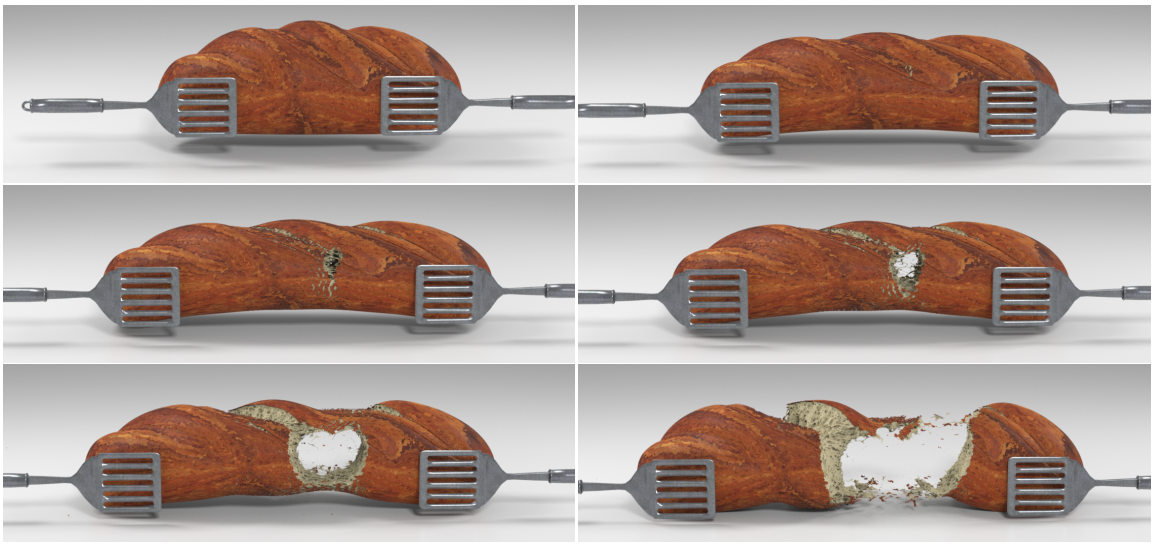


Fig. 13. Our method produces the intricate fracture patterns that result from the tearing of a loaf of bread. We show rendered frames from the simulation (to be seen left to right, top to bottom). The loaf model has around 620k tetrahedra.

5.1.1 Ductile Fracture. We illustrate the highly detailed and intricate fracture effects produced by tearing a loaf of bread in Figure 1 and Figure 13. It is evident from these figures that our simulation model can model complex fracture patterns and can easily scale to high resolution meshes. In Figure 2 we simulate the limbs of a jello armadillo being ripped off. Here we also visualize the corresponding strain profiles. In the frame shown on the leftmost side, the armadillo appears just prior to the fracture. Thus, it has the highest strain on elements that are going to be damaged (coloured red). Once damage sets in post-fracture, as shown on the right, the strain decreases again. The disconnected nodes in the damaged mesh have very low strain values and thus appear in blue-green. In Figure 6, we show the tearing of a doormat with different sizes of kernel support, R_d, thus producing localized (middle) and diffused (bottom) fracture.

5.1.2 Brittle Fracture. For brittle fracture simulation, over our regular graph-based FEM, we have used a stress-based fracture surface initiation and propagation algorithm similar to [Koschier et al. 2015] [Glondu et al. 2013]. Similar to those works, first we identify a few regions containing the highest amount of stress and then initiate a single fracture surface from each of these regions. Each of these fracture surfaces propagates to the boundary of the object. We first calculate the intersections of a fracture surface with the edges of our FEM mesh and then mark these edges as damaged with the embedded intersection points. Finally, the graph-based FEM is used to simulate the fractured object.

In the top row of Figure 3, a hammer hits an armadillo made of solid blue jade to generate large chunks of debris from the fracture. In the bottom row of the same figure, a hammer smashes a porcelain shell model of an armadillo.

Using our fracture simulation method, in the top row of Figure 4 we render the fracture of a solid glass model of a bunny when it is hit by a hammer. The fracture of a shell model of a bunny made of porcelain is simulated in the bottom row of the same figure.

5.2 Effect of Mesh Resolution

We show the effect of mesh resolution on the fracture simulation in Figure 14. In the figure, the computational mesh resolution is increased from top to bottom with topmost, middle and bottom-most models consisting of 1.5k, 38k and 620k tetrahedra respectively. As evident from the figure, with the increase of mesh resolution, finer and more intricate details of the fracture can be captured. The simulation times required are 0.013 sec/frame, 0.31 sec/frame and 4.38 sec/frame respectively from lowest to highest resolution mesh.

5.3 Debris control

We can control the amount of debris produced due to fracture by tuning different parameters. In Figure 15, a bread fracture are shown with varying amount of debris. The parameters that we have tuned in this example to produce less debris are plasticity and fracture threshold. The image on the left has plasticity and fracture threshold $2.5e+06$ Pa and $3.0e+06$ Pa respectively. On the contrary, the plasticity and fracture threshold for the image on right is $1.0e+06$ Pa and $4.0e+06$ Pa respectively.

For brittle fracture, as discussed earlier, first we pick out a few regions containing the highest amount of stress and then initiate a single fracture surface from each of these regions. Now, if more number of high stress regions are picked out, the amount of debris also increases. In Figure 16, the amount of debris for the brittle fracture of a porcelain bunny shell is depicted. In the figure, we choose 50 high stress regions for the leftmost image, 150 high stress regions for the middle image and 300 high stress regions for the rightmost image. It can be seen in the figure that the amount of debris increases from left to right with the increasing number of high stress regions. The user can easily change this parameter to control debris.

5.4 Experimental Validation

To evaluate and validate how faithfully real-world fracture can be simulated using our method, we present a qualitative and quantitative comparison with three benchmark laboratory fracture experiments. These are the Charpy impact test, the Izod impact test and the Brazilian test.

5.4.1 Charpy Impact Test. In this test, we use a steel specimen of dimension $10mm \times 10mm \times 55mm$ with a 45° 'V' shaped groove in the middle of it (see the left image of Figure 17). A pendulum of known mass and length then impacts the back of the groove with both ends of the specimen held against fixed beams. Now depending on the tensile strength

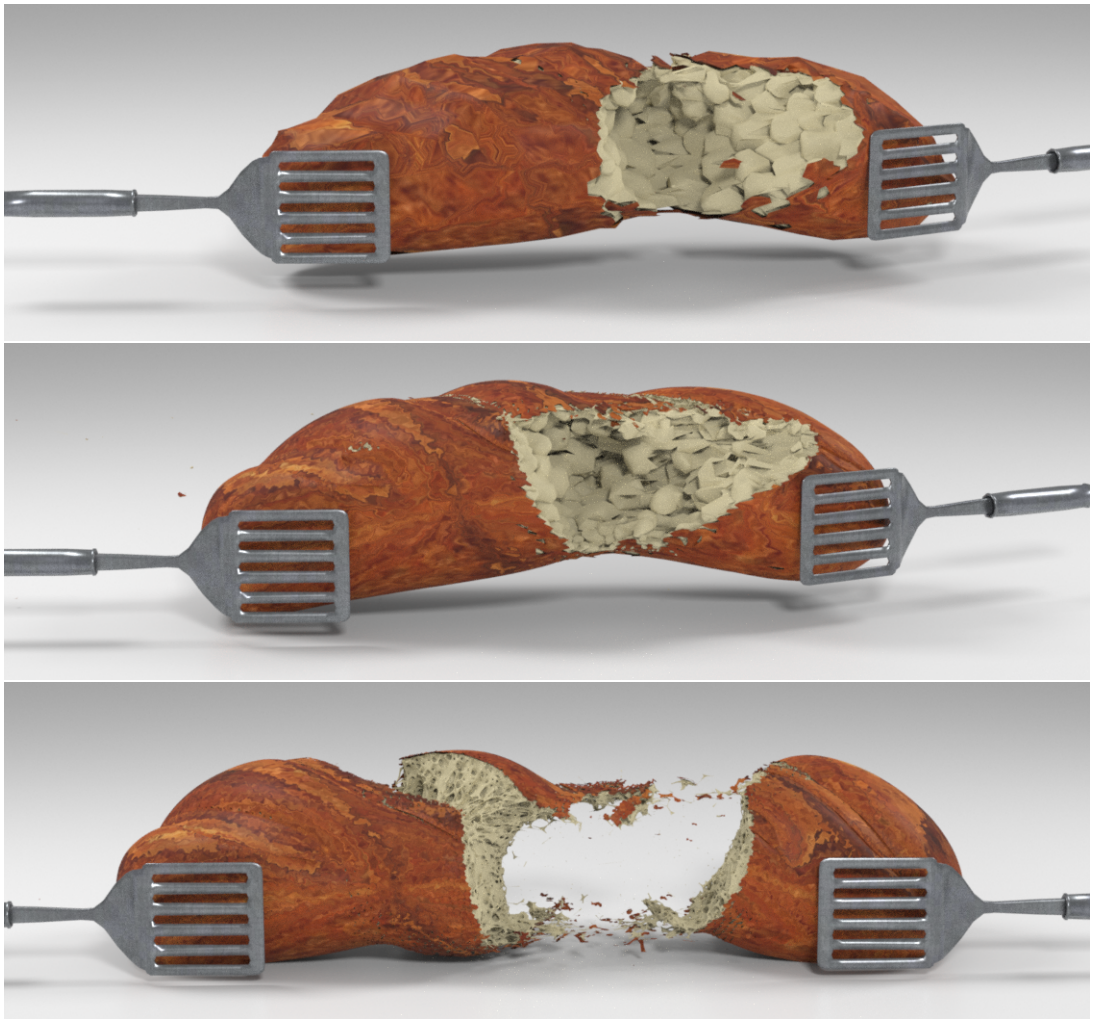


Fig. 14. Computational mesh resolution is increased from top to bottom.

Fig. 15. Fracture simulation with varying amount of debris. The left column shows more debris while the right column shows less amount of debris.

and plasticity threshold of the material, the specimen breaks fully into two pieces or is partially damaged and bends in a three-point configuration. We simulate this experiment using our method, on specimens made of two different kinds

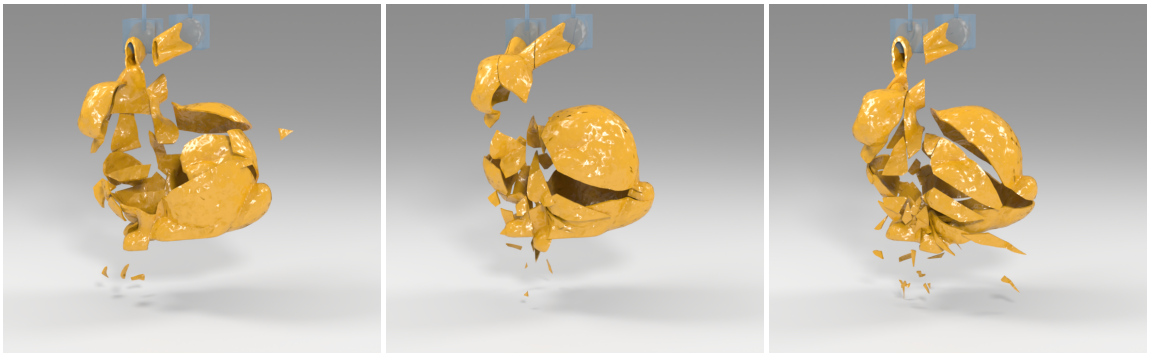


Fig. 16. Fracture simulation with increasing (left to right) amount of debris.

Fig. 17. **Charpy Impact Test**: The left image shows the configuration of the Charpy test. In the middle image, a specimen with a higher plasticity threshold splits into two, while on the right a specimen with a low plasticity threshold only bends and gets partially damaged.

of steel with different tensile strengths (the name of the steel is mentioned in Table 2). Instead of the entire swinging pendulum, as present in the real Charpy test setup, we simulate an impact with a fast-moving block (as can be seen in the image) that hits the specimen at the same location where the specimen is supposed to be hit in the real test. As shown in Figure 17 the steel with a higher plasticity threshold (middle image) breaks into two pieces while the other one (right image) splits partially and bends. Simulation material parameters used in this experiment are obtained from [Tronskar et al. 2002].

5.4.2 Izod Impact Test. The Izod impact test specimen is shown in the left image of Figure 19. Here the specimen is held in a cantilever beam configuration with one end fixed and the pendulum hits it on the other end. The results of the Izod test performed on the same materials as before are shown in the middle and right images of the figure.

In Figure 18 and Figure 20 we plot the load-displacement curves for the Charpy and Izod tests respectively as obtained from our simulated experiments. The solid blue curve denotes simulated brittle fracture, which splits sharply beyond a load threshold. The solid red curve which denotes simulated ductile fracture indicates a plastic flow in the material. The brittle and ductile fracture curves for both tests closely resemble the ground truth reference force-displacement curves shown in the same figure with dotted lines. The ground truth reference curves are reproduced from the same experiments reported in [Tronskar et al. 2002]. The reference data for ductile fracture is obtained from Figure 10

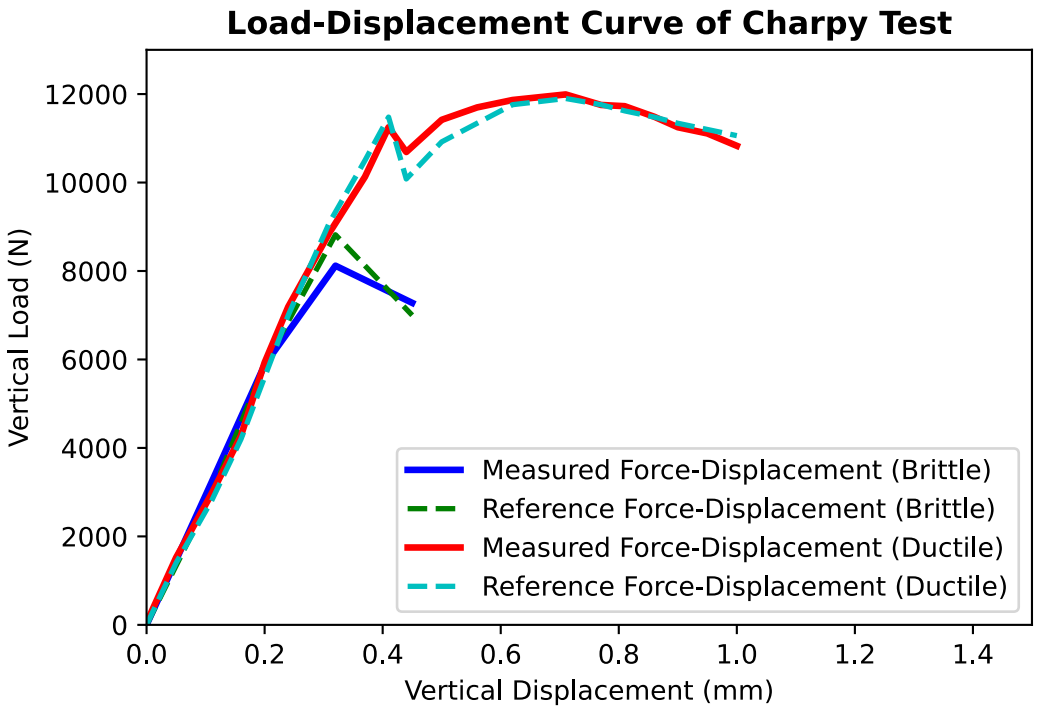


Fig. 18. The plot shows the load-displacement curves of our simulated Charpy impact test. The simulated curves are compared with ground truth curves from actual laboratory experiments.

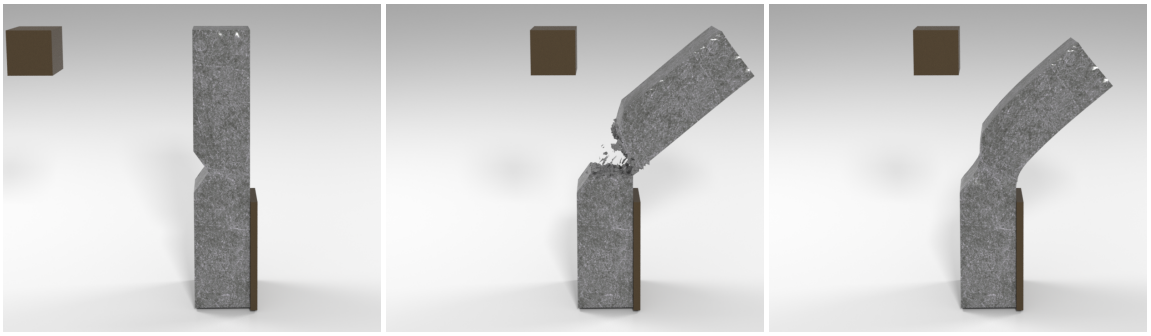


Fig. 19. **Izod Impact Test**: The left image shows the configuration of the Izod test. In the middle image, a specimen with a higher plasticity threshold splits into two, while on the right a specimen with a low plasticity threshold only bends and gets partially damaged.

in [Tronskar et al. 2002]. In this figure, both the force/load and load-line displacement are plotted against the time axis. However, in our plots, we depicted the force/load curve against the load-line displacement. Because in standard practice, the ductility and brittleness of material are denoted using the load-displacement curve as illustrated in Figure

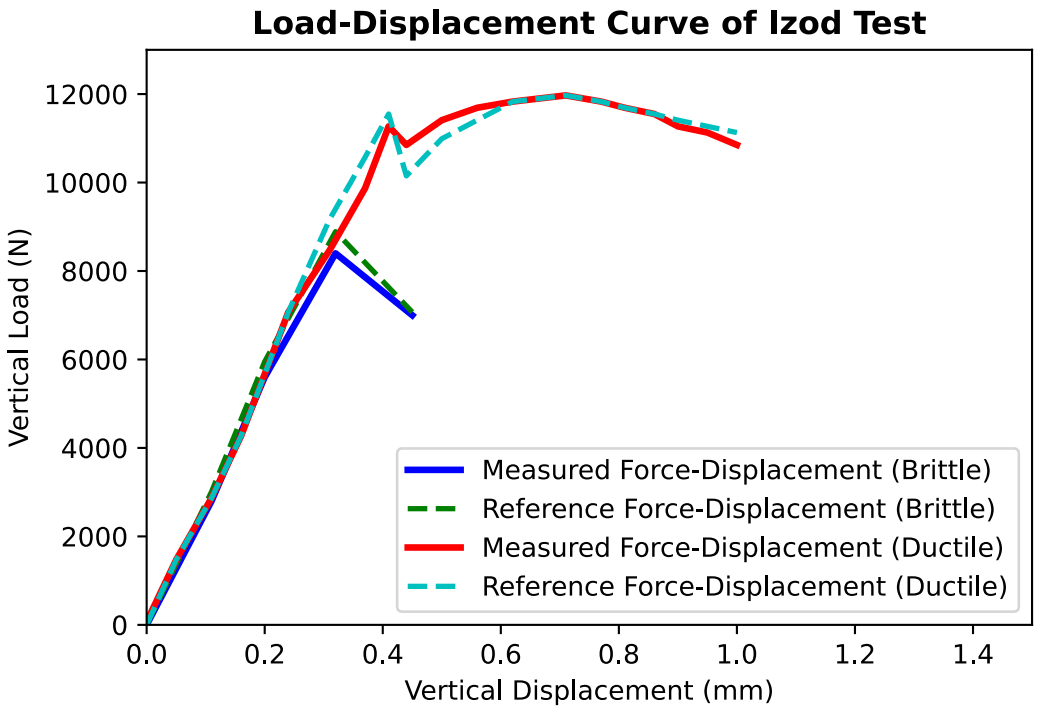


Fig. 20. The plot shows the load-displacement curves of our simulated Izod impact test. The simulated curves are compared with ground truth curves from actual laboratory experiments.

4 in [Tronskar et al. 2002]. Similarly, the reference data for brittle fracture is obtained from Figure 12 in [Tronskar et al. 2002]. Brittle material fractures completely after the elastic region whereas ductile material partially fractures and then bends due to plasticity. This proves that brittle and ductile material simulation performed by our method closely matches the behaviour of real-world materials, both qualitatively and quantitatively.

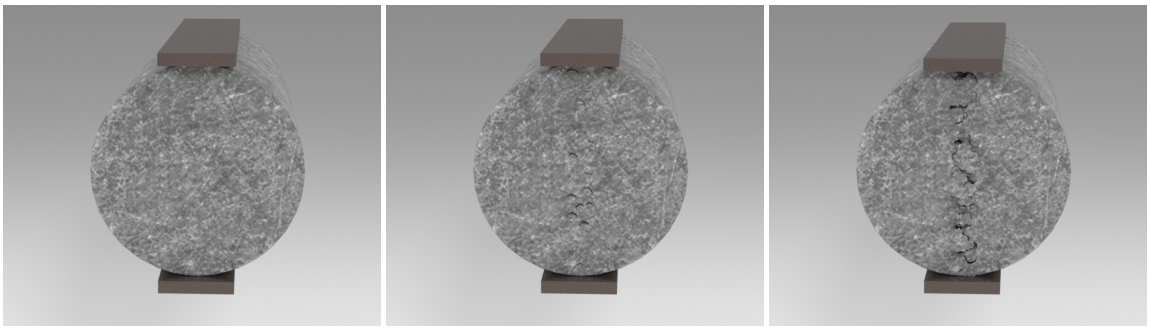


Fig. 21. **Brazilian test**: Initial (left) and fractured (middle & right) configuration of cement cylinder.

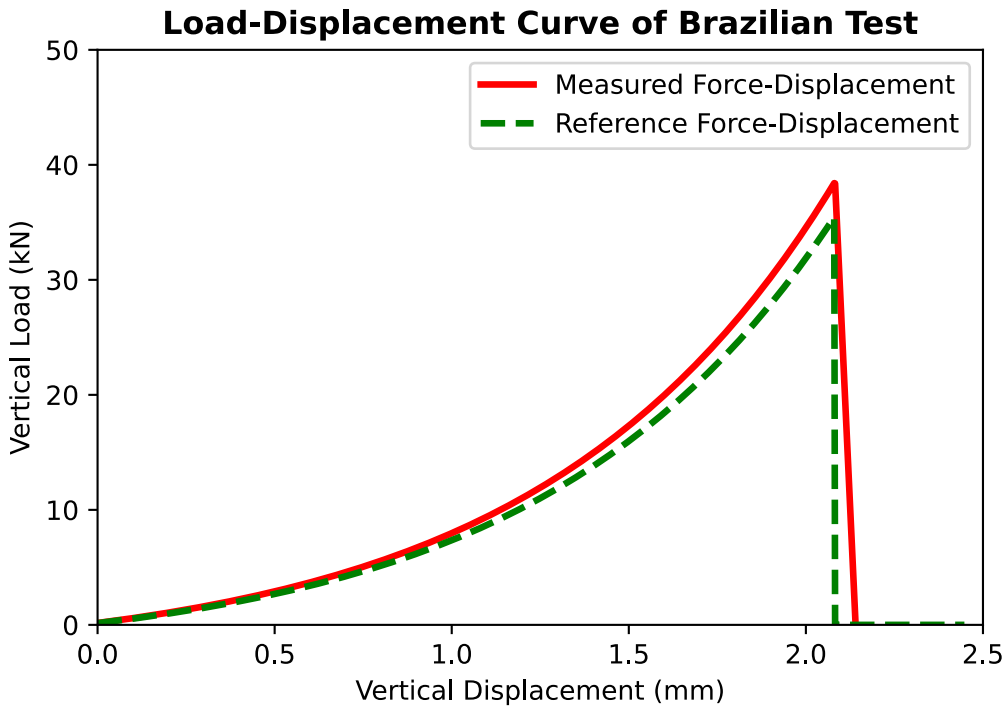


Fig. 22. **Brazilian test**: Load displacement curve for original and simulated experiments.

5.4.3 Brazilian Test. The Brazilian test is a laboratory test for the indirect measurement of tensile strength. In this test, a vertical load is applied at the highest point of a cylinder, the axis of which is placed horizontally and is supported on a horizontal plane. Here we use a cylinder made of granite. In Figure 21 the initial (left) and fractured (middle & right) configurations of the cylinder are shown. We compare the precision of the experiment performed in simulation using our method with a similar real experiment from literature [Ghouli et al. 2021] [ExpeditionWorkshed 2013] by plotting their corresponding load-displacement curves in Figure 22. The close match between the two curves validates the accuracy of our simulation model. Simulation material parameters used in this experiment are obtained from [Ghouli et al. 2021].

5.5 Comparison with Existing Techniques

To the best of our knowledge, this is the first work in computer graphics that presents a FEM-based fracture method that is graph-based, remeshing-free, highly stable and can incorporate a large number of cracks with little extra computational overhead on FEM. We have compared our method with other existing fracture simulation algorithms in the literature.

5.5.1 XFEM vs Our method. Unlike XFEM [Koschier et al. 2017] [Chitalu et al. 2020] or VNA [Sifakis et al. 2007] [Wang et al. 2015], the size of the system matrix remains constant throughout the simulation, thus reducing the computational

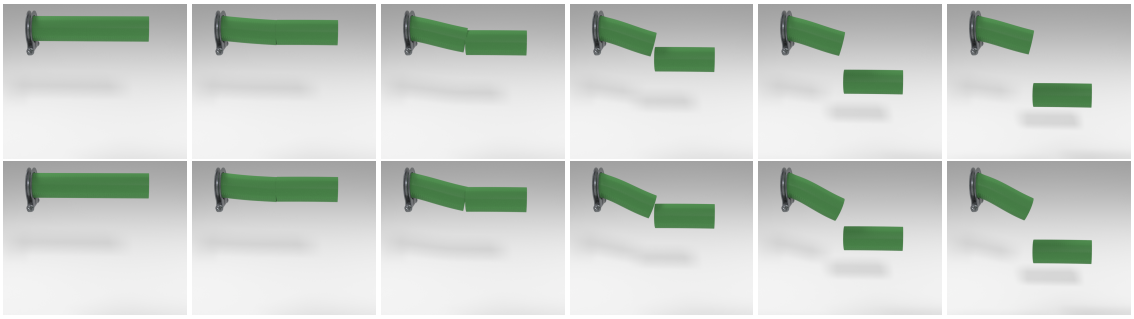


Fig. 23. **XFEM vs Our Method**: A cylinder is split with a single cut. This is simulated using XFEM (top row) and Graph-based FEM (bottom row). Graph-based FEM produces comparable results with significantly lower run-time.

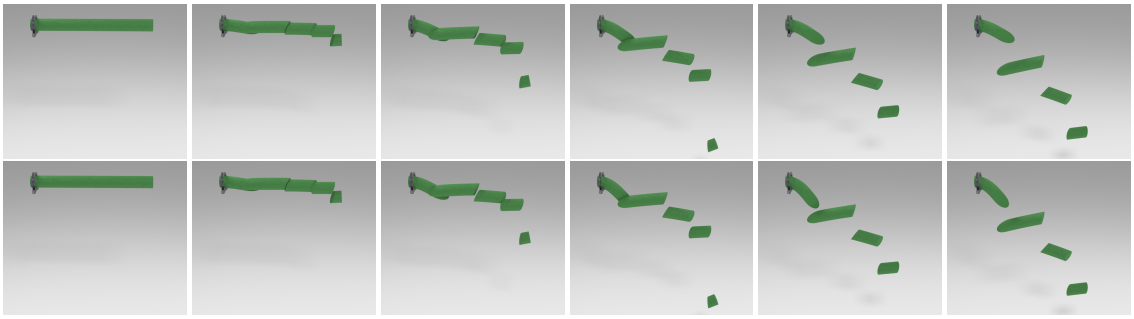


Fig. 24. **XFEM vs Our Method**: A cylinder is split into many parts with multiple random cuts. This is simulated using XFEM (top row) and Graph-based FEM (bottom row).

cost over standard FEM substantially. We have implemented the method presented by Koschier et al. [Koschier et al. 2017] and then compared it with our method. The visual comparison between XFEM and graph-based FEM is presented in Figure 23. As shown in the figure, we use a cylindrical object consisting of 600 DOFs (nodes) and 1566 tetrahedra with a pre-defined vertical split in the middle. To simulate the split, we need to enrich appropriate nodes for XFEM and relabel the appropriate edges as damaged for our graph-based FEM simulation. After enrichment, the number of DOFs (original nodes and enriched nodes together) for XFEM simulation increases to 852 while the number of DOFs remains the same in graph-based FEM. Next, we let the disjoint piece fall under gravity and calculate the average frame rate for both simulations. The simulation of XFEM takes around 0.271 sec/frame while our graph-based FEM requires 0.009 sec/frame. Similarly, in Figure 24, we use another cylindrical object consisting of 7.7k DOFs and 33.5k tetrahedra with multiple random splits. After enrichment, the number of DOFs for XFEM simulation increases to 9.2k while the number of DOFs remains the same in graph-based FEM. The simulation of XFEM takes around 5.87 sec/frame while our graph-based FEM requires 0.253 sec/frame.

We also present a numerical comparison between our method and XFEM-based fracture simulation by Chitalu et al. [Chitalu et al. 2020]. Using their open source code-base [Chitalu 2020], the XFEM-based method requires 2743 sec to render 29 number of cuts for a solid bunny model consisting of 23.7k tetrahedra. On the other hand, our graph-based FEM model requires only 0.18 sec to render the same number of cuts on the same bunny model. Thus, we have a clear advantage in terms of the speed of simulation. Moreover, simulating a large amount of debris with existing FEM or

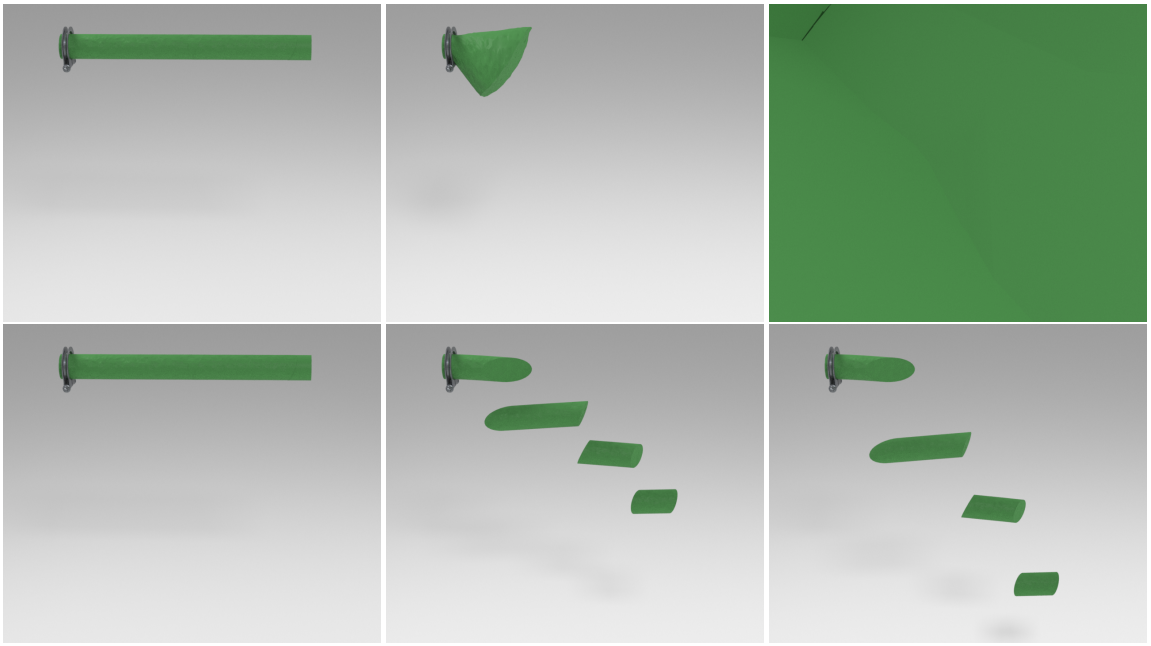


Fig. 25. **Stability of XFEM vs Our Method**: Multiple random splits of a cylindrical object are simulated using XFEM (top row) and Graph-based FEM (bottom row). For a larger time step ($\Delta t = 5 \times 10^{-2}$sec) XFEM simulation becomes unstable while our method remains stable.

XFEM-based solutions in literature requires prohibitively expensive numerical computation due to remeshing and linear scaling of the system matrix. Our method can simulate a very high amount of fracture debris easily. Thus, our method enjoys the good characteristics of both FEM and XFEM methods, without the overheads.

Moreover, while rendering multiple random cuts using XFEM, we have noticed that quite often it generates skewed volume ratios of the tetrahedral elements. This in turn makes the system matrix degenerate with high condition number. Thus for a larger time step ($\Delta t = 5 \times 10^{-2}$sec) XFEM quickly becomes unstable (see Figure 25). On the contrary, as the system matrix remains unchanged throughout the simulation for graph-based FEM, our method can remain stable even for a considerably larger time step.

5.5.2 Remeshing-based FEM vs Our method. As no open source implementation of the work [O'Brien and Hodgins 1999] that simulates fracture using FEM followed by appropriate remeshing, is available, we compare our method to it by reproducing the fracture based on the numerical data as reported in the paper. Brittle fracture of an adobe wall, which contains 338 nodes and 1109 elements initially, is rendered in [O'Brien and Hodgins 1999]. After the fracture, the mesh size grows to 6892 nodes and 8275 elements in the final configuration with multiple disjoint segments. Plugging in the same set of parameters as reported in [O'Brien and Hodgins 1999], Vega FEM takes $0.11 - 0.14$ sec/frame to simulate a similar number of disjoint objects, which have an equal number of nodes and elements together. Compared to this, our method can simulate the fracture of an object to produce the similar number of disjoint pieces in just 0.007 sec. Our object mesh consists of approximately 400 nodes and 1200 elements with the same set of material parameters as before. In our graph-based FEM, the number of nodes and tetrahedra remain unchanged even after fracture, resulting

in a speed-up in computation time compared to remeshing-based methods. This shows a reduction by nearly 15 to 20 times in required simulation time using our method.

6 DISCUSSIONS AND FUTURE WORK

We present a novel graph-based remeshing-free FEM approach for ductile and brittle fracture. We derive theoretical proof to show that our method extends to non-linear hyper-elastic strain energy densities. We follow this with the complete algorithmic description of our model. The high stability, speed and robustness of our method are illustrated and established via multiple dynamic experiments with different materials. We evaluate the appeal and realism of our fracture simulations through results on objects made of a wide variety of materials. Comparing with benchmark fracture experiments, we validate the correctness and accuracy of our method. We also compare with existing XFEM simulation methods, providing quantitative and visual evidence about the better performance of our approach.

Even though we believe that our method is capable of producing particularly realistic fracture of a variety of materials at a high speed, it does have some limitations in its current form. Currently, an edge of an element in the object mesh can split into a maximum of two parts, thus limiting the fracture of tetrahedral elements to a maximum of four parts. We wish to extend our work to incorporate a high number of fracture segments in a single element. Moreover, even though our surface remeshing algorithm uses floating point operations for element splitting, we do not employ any safeguard to resolve rare errors that might occur due to floating point operations. Despite this, our algorithm is very stable, even when simulating highly complex fracture patterns. Some possible solutions could be employing the methods from [Wang et al. 2015] [Wang et al. 2020] where authors resolve floating point errors by providing an appropriate error bound or by looking for increasingly tighter sub-intervals. We would like to model anisotropic fractures in future work.

REFERENCES

Hanen Amor, Jean-Jacques Marigo, and Corrado Maurini. 2009. Regularized formulation of the variational brittle fracture with unilateral contact: Numerical experiments. *Journal of the Mechanics and Physics of Solids* 57, 8 (2009), 1209–1229. https://doi.org/10.1016/j.jmps.2009.04.011

K. Aoki, N. H. Dong, T. Kaneko, and S. Kuriyama. 2004. Physically based simulation of cracks on drying 3D solids. In *Proceedings Computer Graphics International, 2004.* IEEE, 357–364.

Zhaosheng Bao, Jeong-Mo Hong, Joseph Teran, and Ronald Fedkiw. 2007. Fracturing Rigid Materials. *IEEE Transactions on Visualization and Computer Graphics* 13, 2 (March 2007), 370–378. https://doi.org/10.1109/TVCG.2007.39

Adam W. Bargteil, Chris Wojtan, Jessica K. Hodgins, and Greg Turk. 2007. A Finite Element Method for Animating Large Viscoplastic Flow. *ACM Trans. Graph.* 26, 3 (July 2007), 16–es. https://doi.org/10.1145/1276377.1276397

Wei Chen, Fei Zhu, Jing Zhao, Sheng Li, and Guoping Wang. 2018. Peridynamics-Based Fracture Animation for Elastoplastic Solids. *Computer Graphics Forum* 37, 1 (2018), 112–124. https://doi.org/10.1111/cgf.13236 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13236

Zhili Chen, Miaojun Yao, Renguo Feng, and Huamin Wang. 2014. Physics-Inspired Adaptive Fracture Refinement. *ACM Trans. Graph.* 33, 4, Article 113 (July 2014), 7 pages. https://doi.org/10.1145/2601097.2601115

Floyd M. Chitalu. 2020. *MCUT*. Retrieved June 10, 2022 from https://github.com/cutdigital/mcut

Floyd M. Chitalu, Qinghai Miao, Kartic Subr, and Taku Komura. 2020. Displacement-Correlated XFEM for Simulating Brittle Fracture. *Computer Graphics Forum* 39, 2 (2020), 569–583. https://doi.org/10.1111/cgf.13953 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13953

Kenny Erleben, Jon Sporring, Knud Henriksen, and Kenrik Dohlman. 2005. *Physics-Based Animation (Graphics Series).* Charles River Media, Inc., USA.

ExpeditionWorkshed. 2013. *Brazilian Test - Tensile Failure of Concrete in Slow Motion.* Retrieved September 2, 2020 from https://www.youtube.com/watch?v=6lkZIrLp_mE

Saeid Ghouli, Bahador Bahrami, Majid R. Ayatollahi, Thomas Driesner, and Morteza Nejati. 2021. Introduction of a Scaling Factor for Fracture Toughness Measurement of Rocks Using the Semi-circular Bend Test. *Rock Mechanics and Rock Engineering* 54, 8 (2021), 4041 – 4058. https://doi.org/10.1007/s00603-021-02468-1

Loeiz Glondu, Maud Marchal, and Georges Dumont. 2013. Real-Time Simulation of Brittle Fracture Using Modal Analysis. *IEEE Transactions on Visualization and Computer Graphics* 19, 2 (2013), 201–209. https://doi.org/10.1109/TVCG.2012.121

David Hahn and Chris Wojtan. 2015. High-Resolution Brittle Fracture Simulation with Boundary Elements. *ACM Trans. Graph.* 34, 4, Article 151 (July 2015), 12 pages. https://doi.org/10.1145/2766896

David Hahn and Chris Wojtan. 2016. Fast Approximations for Boundary Element Based Brittle Fracture Simulation. *ACM Trans. Graph.* 35, 4, Article 104 (July 2016), 11 pages. https://doi.org/10.1145/2897824.2925902

Koichi Hirota, Yasuyuki Tanoue, and Toyohisa Kaneko. 1998. Generation of crack patterns with a physical model. *The Visual Computer* 14 (1998), 126 – 137. https://doi.org/10.1007/s003710050128

Koichi Hirota, Yasuyuki Tanoue, and Toyohisa Kaneko. 2000. Simulation of three-dimensional cracks. *The Visual Computer* 16 (2000), 371 – 378. https://doi.org/10.1007/s003710000069

Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. 2018a. A Moving Least Squares Material Point Method with Displacement Discontinuity and Two-Way Rigid Body Coupling. *ACM Trans. Graph.* 37, 4, Article 150 (July 2018), 14 pages. https://doi.org/10.1145/3197517.3201293

Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018b. Tetrahedral Meshing in the Wild. *ACM Trans. Graph.* 37, 4, Article 60 (July 2018), 14 pages. https://doi.org/10.1145/3197517.3201353

Doug L. James and Dinesh K. Pai. 1999. ArtDefo: Accurate Real Time Deformable Objects. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. ACM Press/Addison-Wesley Publishing Co., USA, 65–72. https://doi.org/10.1145/311535.311542

Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. 2016. The Material Point Method for Simulating Continuum Materials. In *ACM SIGGRAPH 2016 Courses* (Anaheim, California) *(SIGGRAPH '16)*. Association for Computing Machinery, New York, NY, USA, Article 24, 52 pages. https://doi.org/10.1145/2897826.2927348

Parisa Khodabakhshi, J. N. Reddy, and Arun Srinivasa. 2016. GraFEA: a graph-based finite element approach for the study of damage and fracture in brittle materials. *Meccanica* 51 (2016), 3129 – 3147. https://doi.org/10.1007/s11012-016-0560-6

Parisa Khodabakhshi, J. N. Reddy, and Arun Srinivasa. 2019. A nonlocal fracture criterion and its effect on the mesh dependency of GraFEA. *Acta Mechanica* 230 (2019), 3593–3612. https://doi.org/10.1007/s00707-019-02479-8

Ryo Kikuuwe, Hiroaki Tabuchi, and Motoji Yamamoto. 2009. An Edge-Based Computationally Efficient Formulation of Saint Venant-Kirchhoff Tetrahedral Finite Elements. *ACM Trans. Graph.* 28, 1, Article 8 (feb 2009), 13 pages. https://doi.org/10.1145/1477926.1477934

Theodore Kim and David Eberle. 2020. Dynamic Deformables: Implementation and Production Practicalities. In *ACM SIGGRAPH 2020 Courses* (Virtual Event, USA) *(SIGGRAPH '20)*. Association for Computing Machinery, New York, NY, USA, Article 23, 182 pages. https://doi.org/10.1145/3388769.3407490

Dan Koschier, Jan Bender, and Nils Thuerey. 2017. Robust EXtended Finite Elements for Complex Cutting of Deformables. *ACM Trans. Graph.* 36, 4, Article 55 (July 2017), 13 pages. https://doi.org/10.1145/3072959.3073666

Dan Koschier, Sebastian Lipponer, and Jan Bender. 2015. Adaptive Tetrahedral Meshes for Brittle Fracture Simulation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Copenhagen, Denmark) *(SCA '14)*. Eurographics Association, Goslar, DEU, 57–66.

J. A. Levine, A. W. Bargteil, C. Corsi, J. Tessendorf, and R. Geist. 2015. A Peridynamic Perspective on Spring-Mass Fracture. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Copenhagen, Denmark) *(SCA '14)*. Eurographics Association, Goslar, DEU, 47–55.

Christian Miehe, Lisa-Marie Schänzel, and Heike Ulmer. 2015. Phase field modeling of fracture in multi-physics problems. Part I. Balance of crack surface and failure criteria for brittle crack propagation in thermo-elastic solids. *Computer Methods in Applied Mechanics and Engineering* 294 (2015), 449–485. https://doi.org/10.1016/j.cma.2014.11.016

Neil Molino, Zhaosheng Bao, and Ron Fedkiw. 2004. A Virtual Node Algorithm for Changing Mesh Topology during Simulation. In *ACM SIGGRAPH 2004 Papers* (Los Angeles, California) *(SIGGRAPH '04)*. Association for Computing Machinery, New York, NY, USA, 385–392. https://doi.org/10.1145/1186562.1015734

Matthias Müller and Markus Gross. 2004. Interactive Virtual Materials. In *Proceedings of Graphics Interface* (London, Ontario, Canada). Canadian Human-Computer Communications Society, Waterloo, CAN, 239–246.

Alan Norton, Greg Turk, Bob Bacon, John Gerth, and Paula Sweeney. 1991. Animation of fracture by physical modeling. *The Visual Computer* 7 (1991), 210 – 219. https://doi.org/10.1007/BF01900837

James F. O'Brien, Adam W. Bargteil, and Jessica K. Hodgins. 2002. Graphical Modeling and Animation of Ductile Fracture. *ACM Trans. Graph.* 21, 3 (July 2002), 291–294. https://doi.org/10.1145/566654.566579

James F. O'Brien and Jessica K. Hodgins. 1999. Graphical Modeling and Animation of Brittle Fracture. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. ACM Press/Addison-Wesley Publishing Co., USA, 137–146. https://doi.org/10.1145/311535.311550

Mark Pauly, Richard Keiser, Bart Adams, Philip Dutré, Markus Gross, and Leonidas J. Guibas. 2005. Meshless Animation of Fracturing Solids. In *ACM SIGGRAPH 2005 Papers* (Los Angeles, California) *(SIGGRAPH '05)*. Association for Computing Machinery, New York, NY, USA, 957–964. https://doi.org/10.1145/1186822.1073296

Tobias Pfaff, Rahul Narain, Juan Miguel de Joya, and James F. O'Brien. 2014. Adaptive Tearing and Cracking of Thin Sheets. *ACM Trans. Graph.* 33, 4, Article 110 (July 2014), 9 pages. https://doi.org/10.1145/2601097.2601132

J.N. Reddy and Arun Srinivasa. 2015. On the force–displacement characteristics of finite elements for elasticity and related problems. *Finite Elements in Analysis and Design* 104 (2015), 35 – 40. https://doi.org/10.1016/j.finel.2015.04.011

Hang Si. 2015. TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator. *ACM Transactions on Mathematical Software (TOMS)* 41, 2, Article 11 (Feb. 2015), 36 pages. https://doi.org/10.1145/2629697

Eftychios Sifakis and Jernej Barbic. 2012. FEM Simulation of 3D Deformable Solids: A Practitioner's Guide to Theory, Discretization and Model Reduction. In *ACM SIGGRAPH 2012 Courses* (Los Angeles, California) *(SIGGRAPH '12)*. Association for Computing Machinery, New York, NY, USA, Article 20,

50 pages. https://doi.org/10.1145/2343483.2343501

Eftychios Sifakis, Kevin G. Der, and Ronald Fedkiw. 2007. Arbitrary Cutting of Deformable Tetrahedralized Objects. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Diego, California) *(SCA '07)*. Eurographics Association, Goslar, DEU, 73–80.

F. S. Sin, D. Schroeder, and J. Barbič. 2013. Vega: Non-Linear FEM Deformable Object Simulator. *Computer Graphics Forum* 32, 1 (2013), 36–48. https://doi.org/10.1111/j.1467-8659.2012.03230.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2012.03230.x

Breannan Smith, Fernando De Goes, and Theodore Kim. 2018. Stable Neo-Hookean Flesh Simulation. *ACM Trans. Graph.* 37, 2, Article 12 (March 2018), 15 pages. https://doi.org/10.1145/3180491

A. R. Srinivasa. 2021. Discrete differential geometry and its role in computational modeling of defects and inelasticity. *Meccanica* 56, 7 (2021). https://doi.org/10.1007/s11012-021-01335-1

Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. 2013. A Material Point Method for Snow Simulation. *ACM Trans. Graph.* 32, 4, Article 102 (July 2013), 10 pages. https://doi.org/10.1145/2461912.2461948

Min Tang, Dinesh Manocha, Miguel A. Otaduy, and Ruofeng Tong. 2012. Continuous Penalty Forces. *ACM Trans. Graph.* 31, 4, Article 107 (July 2012), 9 pages. https://doi.org/10.1145/2185520.2185603

Min Tang, Dinesh Manocha, and Ruofeng Tong. 2010. Fast Continuous Collision Detection Using Deforming Non-Penetration Filters. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (Washington, D.C.) *(I3D '10)*. Association for Computing Machinery, New York, NY, USA, 7–13. https://doi.org/10.1145/1730804.1730806

Demetri Terzopoulos and Kurt Fleischer. 1988. Modeling Inelastic Deformation: Viscolelasticity, Plasticity, Fracture. *SIGGRAPH Comput. Graph.* 22, 4 (June 1988), 269–278. https://doi.org/10.1145/378456.378522

J.P Tronskar, M.A Mannan, and M.O Lai. 2002. Measurement of fracture initiation toughness and crack resistance in instrumented Charpy impact testing. *Engineering Fracture Mechanics* 69, 3 (2002), 321 – 338. https://doi.org/10.1016/S0013-7944(01)00077-7

Bolun Wang, Zachary Ferguson, Teseo Schneider, Xin Jiang, Marco Attene, and Daniele Panozzo. 2020. A Large Scale Benchmark and an Inclusion-Based Algorithm for Continuous Collision Detection. arXiv:2009.13349 [cs.GR]

Stephanie Wang, Mengyuan Ding, Theodore F. Gast, Leyi Zhu, Steven Gagniere, Chenfanfu Jiang, and Joseph M. Teran. 2019. Simulation and Visualization of Ductile Fracture with the Material Point Method. *Proc. ACM Comput. Graph. Interact. Tech.* 2, 2, Article 18 (July 2019), 20 pages. https://doi.org/10.1145/3340259

Yuting Wang, Chenfanfu Jiang, Craig Schroeder, and Joseph Teran. 2015. An Adaptive Virtual Node Algorithm with Robust Mesh Cutting. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Copenhagen, Denmark) *(SCA '14)*. Eurographics Association, Goslar, DEU, 77–85.

Jeffrey A. Weiss, Bradley N. Maker, and Sanjay Govindjee. 1996. Finite element implementation of incompressible, transversely isotropic hyperelasticity. *Computer Methods in Applied Mechanics and Engineering* 135, 1 (1996), 107–128. https://doi.org/10.1016/0045-7825(96)01035-3

Martin Wicke, Daniel Ritchie, Bryan M. Klingner, Sebastian Burke, Jonathan R. Shewchuk, and James F. O'Brien. 2010. Dynamic Local Remeshing for Elastoplastic Simulation. In *ACM SIGGRAPH 2010 Papers* (Los Angeles, California) *(SIGGRAPH '10)*. Association for Computing Machinery, New York, NY, USA, Article 49, 11 pages. https://doi.org/10.1145/1833349.1778786

Joshuah Wolper, Yunuo Chen, Minchen Li, Yu Fang, Ziyin Qu, Jiecong Lu, Meggie Cheng, and Chenfanfu Jiang. 2020. AnisoMPM: Animating Anisotropic Damage Mechanics. *ACM Trans. Graph.* 39, 4, Article 37 (July 2020), 16 pages. https://doi.org/10.1145/3386569.3392428

Joshuah Wolper, Yu Fang, Minchen Li, Jiecong Lu, Ming Gao, and Chenfanfu Jiang. 2019. CD-MPM: Continuum Damage Material Point Methods for Dynamic Fracture Animation. *ACM Trans. Graph.* 38, 4, Article 119 (July 2019), 15 pages. https://doi.org/10.1145/3306346.3322949

Yufeng Zhu, Robert Bridson, and Chen Greif. 2015. Simulating Rigid Body Fracture with Surface Meshes. *ACM Trans. Graph.* 34, 4, Article 150 (July 2015), 11 pages. https://doi.org/10.1145/2766942