

# Displacement-Correlated XFEM for Simulating Brittle Fracture

Floyd M. Chitalu<sup>1</sup> Qinghai Miao<sup>2†</sup> Kartic Subr<sup>1</sup> Taku Komura<sup>1</sup>

<sup>1</sup>University of Edinburgh

<sup>2</sup>University of Chinese Academy of Sciences



Figure 1: Our method offers a high-resolution crack propagation scheme on an explicit surface mesh without the need for adaptively tetrahedralized input meshes. We achieve this by combining a novel adaptation of the extended finite element method (XFEM), and a polygon-based cutting algorithm to compute fragments which retain their characteristic ridge-like structures—and sharpness—due to cracks. The figure shows an impact between a statue of The Winged Victory of Samothrace and a wrecking ball.

## Abstract

We present a remeshing-free brittle fracture simulation method under the assumption of quasi-static linear elastic fracture mechanics (LEFM). To achieve this, we devise two algorithms. First, we develop an approximate volumetric simulation, based on the extended Finite Element Method (XFEM), to initialize and propagate Lagrangian crack-fronts. We model the geometry of fracture explicitly as a surface mesh, which allows us to generate high-resolution crack surfaces that are decoupled from the resolution of the deformation mesh. Our second contribution is a mesh cutting algorithm, which produces fragments of the input mesh using the fracture surface. We do this by directly operating on the half-edge data structures of two surface meshes, which enables us to cut general surface meshes including those of concave polyhedra and meshes with abutting concave polygons. Since we avoid triangulation for cutting, the connectivity of the resulting fragments is identical to the (uncut) input mesh except at edges introduced by the cut. We evaluate our simulation and cutting algorithms and show that they outperform state-of-the-art approaches both qualitatively and quantitatively.

## CCS Concepts

• **Computing methodologies** → **Physical simulation**; **Mesh geometry models**;

## 1. Introduction

Realistic depiction of breaking objects is desirable across a range of computer graphics applications including special effects, computer animation and video games. The breakage of 3D models can be mimicked either manually via tedious artistic specification or using

automatic algorithms. The latter can be classified into methods that exploit *heuristics* [SSF09; MCK13] and those that compute physical simulations of the mechanics of fracture. Simulation methods typically have origins in engineering and are therefore designed to be accurate for targeted applications. Adapting them to suit computer graphics applications often requires overcoming challenges such as reducing computational cost and generalizing to realistic boundary conditions [HWI6] as well as providing controllability [CYFW14].

† Floyd M. Chitalu and Qinghai Miao are joint first authors.

Despite the effort of computer graphics researchers, existing techniques still suffer from either scalability, stability or realism problems for simulating the propagation of crack surfaces on arbitrary shapes. Classic finite element method (FEM)-based approaches [OH99] require conforming the domain mesh to the crack surfaces by re-meshing, which poses several challenges when treating evolving fractures. To avoid the cost and instability caused by re-meshing, some methods operate on particle systems [PKA\*05; SSC\*13; WFL\*19]. These meshless methods introduce difficulties with respect to enforcing essential boundary conditions, computational cost and overall rigidity of computed fragments. Boundary element method (BEM)-based methods use surface representations [HW15; ZBG15; HW16]. In contrast to other discretization methods (e.g. FEM), these methods involve singular integrals which can be prohibitively expensive to solve and are restricted to materials characterised by large volumes. Also, they are not readily able to seed cracks on the interior due to their boundary integral formulation.

The extended finite element method (XFEM) [MDB99; Do199; DMD\*00] is proposed for decoupling the simulation mesh from the crack surface: however, prominent methods that represent the crack surfaces by level-sets require using high resolution simulation meshes for simulating detailed crack surfaces. Also, tracking the dynamic propagation of the crack surface by level sets is inherently difficult.

In this paper, we present an efficient brittle fracture simulation method in high resolution and without any requirement on the simulation mesh. In this method, we combine XFEM with a high-resolution crack propagation scheme on an explicit surface mesh, resulting in efficient framework for brittle fracture. Our method allows for handling crack propagation without re-meshing (changing the simulation mesh) or using crack-tip enrichment, which has several advantages including reducing the number of degrees of freedom (DOF). Using a volumetric setting, we can simulate brittle fracture on a broad range of domains and accommodate spatially varying material parameters to control fracture. To represent a crack we adopt an explicit approach which can simplify the procedure to propagating the crack surface within the volumetric domain. To cope with the geometric operations of cutting meshes, we propose a novel algorithm that extends the approach by Sifakis *et al.* [SDF07] but copes with concavities, without needing triangulation.

We simulate results showing detailed and realistic fracture of multiple brittle objects colliding and breaking into small pieces (see Fig. 1, and § 7). Our method also allows animators to control the breakage by biasing the toughness within the domain, and to control the distinctive look sought from real-world materials in a simple fashion. The generality of our cutting algorithm makes it applicable to a wide number of use-cases. It is a simple and robust approach to perform various operations (*e.g.* boolean operations) on open and closed surfaces, making it useful for cases within—and even beyond—the scope of immediate application. Further, as we seek to minimise numerical operations for mesh extraction, our algorithm resolves all mesh connectivity using only combinatorial structure except when computing polygon intersections.

Fig. 2 provides a visual overview of existing approaches. Our contributions may be summarized as follows:

- A novel quasi-static brittle fracture simulation method on the basis of XFEM but without crack-tip enrichment, and
- a general and suitably robust implementation for explicit surface mesh cutting, using arbitrary polygonal subdivisions.

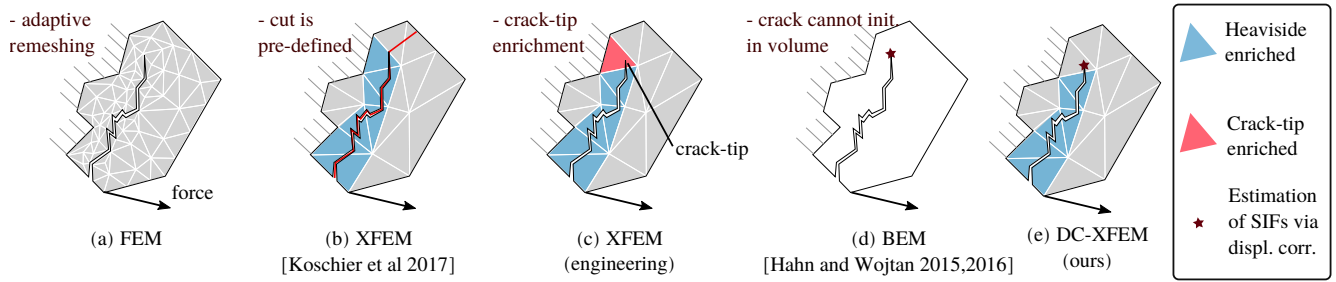
## 2. Related Work

In this section, we first review methods for simulating fracture and re-meshing in computer graphics. Next, we briefly review XFEM as presented in engineering literature. Finally, we review methods for tackling the challenging task of mesh cutting, which is common in fracture simulation.

**Fracture Simulation in Computer Graphics:** Physically-based fracture is a well studied problem in computer graphics, stemming from the seminal work by Terzopoulos *et al.* [TF88]. Early approaches proposed mass-spring systems to model brittle fracture with stress-based yield thresholds [NTB\*91; ADKK04]. However, visual artifacts were common due to spring removal and representing crack surfaces was non-trivial. Approaches based on FEM have had wider success [OH99; O'B02; SWB01; PO09; KLB14; BHTF07; MG04; MMDJ01]. The earliest of these used nodal stress analysis to perform planar fracture for brittle and ductile material settings [OH99; O'B02]. O'Brien and Hodgins [OH99] introduced brittle fracture with FEM which was later extended by others [BHTF07; KLB14; MG04]. Bao *et al.* [BHTF07] also present a method for simulating both brittle and ductile (denting) fracture. A real-time method for brittle fracture is presented Parker and O'Brien [PO09] which is based on O'Brien and Hodgins' method [OH99] but with refinement procedures to ensure that the meshes stayed self-consistent. Glondu *et al.* [GMD13] also present a real-time approach with FEM, using modal analysis to handle crack initiation and an energy-driven algorithm for fracture propagation on implicit surfaces.

Re-meshing is frequently used in FEM simulation to handle high stress distributions accurately; align tetrahedral meshes with cracks; or to simply overcome fracture resolution constraints. Wick *et al.* [WRK\*10] use dynamic local mesh refinement (and coarsening) to repair degraded tetrahedra, while Chen *et al.* [CYFW14] handle remeshing based on gradient descent flow to enhance fracture resolution and detail. Koschier *et al.* [KLB14] also present an adaptive subdivision scheme to facilitate cutting during fine breakage on the basis of the virtual node algorithm (VNA) [MBF05].

The challenges of topological discontinuities via re-meshing can be addressed through methods such as Discontinuous Galerkin FEM (DGFEM) [KMBG08] which requires moving-least squares interpolation. The material point method (MPM) has also been used with level sets to simulate brittle and ductile fracture [HJST13; WFL\*19]. Though impressive, MPM may preclude the ability to represent of infinitely sharp features, and with difficulties handling rigid shatter effects. XFEM embedding [KMB\*09] improves accuracy but imposes limits on the fracture geometry. It scales poorly with the resolution of the fracture surface. Richardson *et al.* [RHS\*09] also present a crack propagation scheme which uses a finer mesh for integration purposes and a geometric mesh cutting tool [SDF07] for full XFEM enrichment but in 2D. A related extension to 3D is described by Koschier *et al.* [KBT17] (see also Jeřábková and



**Figure 2:** An illustrated comparison of relevant work.

Kuhlen [JK09]) but using only Heaviside enrichment to simulate the dynamics of meshes with cuts, since specifying of crack-tip enrichment is non-trivial in 3D. Their work also does not address the generation or propagation of fracture, so the cuts are pre-specified.

Some methods simulate fracture with surface meshes. For example, Pfaff *et al.* [PNJO14] capture the tearing of thin sheets by solving the elasto-plastic equations on a triangulated finite element mesh. Zhu *et al.* [ZBG15] simulate brittle fracture based on a boundary integral formulation of elasticity combined with mesh evolution and a rigid body solver from which contact forces are extracted as Neumann boundary conditions. Hahn and Wojtan [HW15] similarly adapted BEM for computing stress on surfaces with spatially varying fracture parameters to produce interesting effects. They estimate stress intensity factors (SIFs) along crack fronts and use these for crack propagation. Their method simulates fracture on a coarse crack surface accompanied by an implicit surface to address the poor scaling properties of BEM due to singular integrals.

**XFEM in Engineering:** Classical FEM requires adaptive meshing to handle evolving discontinuities in the simulation domain, such as in the case of fracture. XFEM was first introduced by Moës *et al.* [MDB99] to address this limitation by introducing discontinuities in the interpolation functions, thereby allowing the simulation mesh to remain unchanged (see also Belytschko and Black [BB99] and others [Do199; DMD\*00]). Mousavi *et al.* [MGS11] present a method to handle multiple intersecting cracks using generalized harmonic enrichment functions but in 2D, which is based on the work of Kaufmann *et al.* [KMB\*09]. XFEM has also been used to model fatigue crack propagation in 3D using planar cracks [SMMB00; SCM03] and curved surfaces [PDGJ09].

Cracks in XFEM were previously modelled purely with *implicit* level set functions [SCMB01]. However, these introduced a tight coupling between the resolution of the simulation mesh and the crack surface, precluding a general ability to incorporate fine details. Alternative *explicit* mesh-based representations of the crack surface have been explored [SCM03; PDGJ09], along with adaptive refinement to accommodate varying propagation speeds along the crack front [GODB13]. These explicit methods are able to model fracture surfaces more realistically and, during crack propagation, it is simpler to update a crack-surface mesh than to update level sets [GODB13; RG17]. Notably, explicit methods also require complicated computational geometric operations between the crack surface and the simulation mesh, which we address. On this surface,

an approach akin to Fries and Baydoun [FB11] is used: Fries and Baydoun propose a hybrid approach that combines the benefits of explicit and implicit representations by inducing the level function using the explicit crack mesh to enrich elements. § 3 presents a review of FEM and XFEM. We refer readers to a textbook [Kho15] for a more thorough treatment.

**Mesh Cutting:** We refer to *cutting* as the process of splitting a mesh into two or more components based on its intersection with a specified cutting surface. One class of algorithms applies piecewise-linear cuts [MBF05; TSB\*05; WJST14; KLB14; PUC\*15]. These algorithms use volumetric decomposition and duplicate vertices along *predefined element faces* that are most aligned with the cutting surface. These methods are relatively simple but are computationally costly for complex cut patterns [PUC\*15; KLB14]. Since they operate using volume-refinement or regular grids [DGW11; JRC14], they suffer from lack of volume preservation and loss of mesh-scale detail [WJST14].

Mesh evolution algorithms can overcome piece-wise linear cutting limitations given their ability to preserve volume, mesh-scale detail, and sharp features [WMFB11; DBG14]. However, these are not directly applicable to fracture. For example, Da *et al.* [DBG14] present a multi-material triangle mesh-based surface tracking scheme for evolving interfaces which has been used for fracture simulation by Zhu *et al.* [ZBG15] but with significant modifications for tracking the crack surface. Sifakis *et al.* [SDF07] also describe a method to cut tetrahedralized meshes with arbitrary incisions, along with novel edge placement rules for reconnecting topology. The method works well but omits details for polygon clipping (“boundary tracking”), and assumes a tetrahedral representation making it applicable only to triangulated meshes.

A third class of approaches converts polygonal meshes to an intermediate sparse voxel representation (*e.g.* OpenVDB [Mus13]) for cutting, before re-triangulation. These require multiple representations of the cutting surface and are useful when the resolution of the cutting surface affects simulation performance as seen in BEM [HW15]. Voxel representations are also used when rendering meshed particles but may require trade-offs between surface smoothness and sharpness [WFL\*19]. A notable advantage of sparse voxel representations is their robustness against numerical error when resolving intersections by using level sets. Conversely, mesh based approaches can be susceptible to unexpected failure from numerical

error if not handled carefully (see e.g. Zhou et al. [ZGZJ16], Wang et al. [WJST14], and Bernstein and Fussell [BF09] for discussions).

We refer readers to Wu et al. [WWD15] for a survey on mesh cutting.

### 3. Review: FEM & XFEM

In this section, we briefly describe the derivation of XFEM on the basis of classical FEM. A similar notation to Khoei [Kho15] and Sukumar [SMMB00] is used, to which we refer interested readers for details.

**Problem:** Consider a body of linear elastic material occupying a domain  $\Omega \subset \mathbb{R}^3$  and subject to loading  $f$ . Under static equilibrium, in the absence of traction forces, it can be derived that

$$\nabla \cdot \boldsymbol{\sigma} = -f, \quad (1)$$

subject to boundary conditions. Here,  $\nabla \cdot$  denotes the divergence operator and  $\boldsymbol{\sigma}$  is cauchy stress which is a function of displacements  $\mathbf{u}(\mathbf{x})$ ,  $\mathbf{x} \in \Omega$ . The goal is to find a function  $\mathbf{u}(\mathbf{x})$  which satisfies the differential equation and boundary conditions.

#### 3.1. FEM Approximation of Linear Elastic Mechanics

FEM reformulates the above *strong form* of the problem as a set of algebraic equations (via a *weak form*) that can be solved numerically. The key difference is that it focuses on finding an unknown vector of displacements  $\mathbf{u}$  rather than a continuous function. The function  $\mathbf{u}(\mathbf{x})$  is then obtained from the vector using relevant interpolation functions known as *shape functions*. The choice of shape functions depends on the discretization of  $\Omega$  into *elements*, and results in a system of algebraic equations.

Assuming each element is a *linear* tetrahedron  $e$  specified by nodes (vertices) and the displacements at the nodes are  $\mathbf{u}_e$ , the interpolated displacement at  $\mathbf{x} \in e$  is

$$\mathbf{u}(\mathbf{x}) = \sum_{i=1}^4 N_i(\mathbf{x}) \mathbf{u}_e^i \quad (2)$$

where  $N_i$  is the shape function of node  $i$  which has displacement  $\mathbf{u}_e^i$ . Strain and stress are constant in  $e$ :

$$\boldsymbol{\varepsilon}_e(\mathbf{x}) = \mathbf{B}_e \mathbf{u}_e; \quad \boldsymbol{\sigma}_e(\mathbf{x}) = \mathbf{D}_e \boldsymbol{\varepsilon}_e(\mathbf{x}). \quad (3)$$

Here  $\mathbf{B}_e$  is the discretized gradient matrix which contains the partial derivatives of the shape functions and  $\mathbf{D}_e$  is the elasticity matrix which encodes material properties.

By the principle of virtual work, variations in internal work must equal variations in external due to boundary constraints (cf. Eq. 2.66 in [Kho15]) which gives the following linear system within each element

$$\mathbf{K}_e \mathbf{u}_e = \mathbf{f}_e \quad \text{where,} \quad \mathbf{K}_e \equiv \int_e \mathbf{B}_e^T \mathbf{D}_e \mathbf{B}_e \, dx, \quad \mathbf{f}_e \equiv \int_e N^T \mathbf{t}_e \, dx. \quad (4)$$

Thus,  $\mathbf{K}_e$  is stiffness matrix obtained from material properties and  $\mathbf{f}_e$  is obtained from external forces  $\mathbf{t}_e$  (boundary conditions) acting on the element. For simulating a mesh with many elements, the element-wise  $\mathbf{K}_e$  from all elements are carefully assembled into

a global sparse linear system  $\mathbf{K}$  to solve for all unknown nodal displacements  $\mathbf{u}$ .

In the presence of fracture, displacements are non-smooth within elements  $e$  intersected by the crack. FEM addresses this problem by aligning the elements with the discontinuity (crack surface). This requires remeshing which needs special care but also results in a larger  $\mathbf{K}$  for high-resolution crack surfaces (more DOFs).

#### 3.2. XFEM for Simulating Fracture

XFEM copes with cracks running through elements by enhancing the approximation space independent of the elements. (Here we consider only *extrinsic* enrichment where more shape functions and unknowns result in the approximation, while operating on the same mesh elements). An element that is cut by a fracture surface may either be completely split or contain the crack-tip within its volume. The former results in a strong discontinuity while the latter leads to singularities in the near-field displacements. The enriched interpolation

$$\mathbf{u}_e^{\text{enr}}(\mathbf{x}) = \sum_{i=1}^m N_i(\mathbf{x}) \mathbf{u}_i + \Upsilon_{\text{hev}}(\mathbf{x}) + \Upsilon_{\text{tip}}(\mathbf{x}), \quad \text{where} \quad (5)$$

$$\Upsilon_{\text{hev}} = \sum_{i=1}^m N_i(\mathbf{x}) (\Psi_{\text{hev}}(\mathbf{x}) - \Psi_{\text{hev}}(\mathbf{x}_i)) \mathbf{a}_i, \quad \text{and} \quad (6)$$

$$\Upsilon_{\text{tip}} = \sum_{i=1}^m \sum_{k=1}^4 N_i(\mathbf{x}) (\Psi_{\text{tip}}(\mathbf{x}) - \Psi_{\text{tip}}(\mathbf{x}_i)) \mathbf{b}_i^k. \quad (7)$$

sums over all  $m$  nodes of the element. For a tetrahedron,  $m = 4$ .  $\Psi_{\text{hev}}(\mathbf{x})$  is the Heaviside (step) function which decouples field quantities (e.g. displacement) on both sides of the crack passing through the element.  $\Psi_{\text{tip}}(\mathbf{x})$  are asymptotic solutions [Wil61] which are specifically chosen in order to capture singularities.  $\mathbf{a}_i$  and  $\mathbf{b}_i^k$  are vectors of added degrees of freedom (DOF) controlling the enrichment. Thus, rather than solving for  $\mathbf{u}_e$  (as in FEM) within each element, the new system  $\mathbf{K}_e^{\text{enr}} \mathbf{u}_e^{\text{enr}} = \mathbf{f}_e^{\text{enr}}$  has additional unknowns:

$$\mathbf{u}_e^{\text{enr}} = [\mathbf{u}_e^T \mathbf{a}^T]^T \quad \text{or} \quad \mathbf{u}_e^{\text{enr}} = [\mathbf{u}_e^T \mathbf{b}_1^T \mathbf{b}_2^T \mathbf{b}_3^T \mathbf{b}_4^T]^T, \quad (8)$$

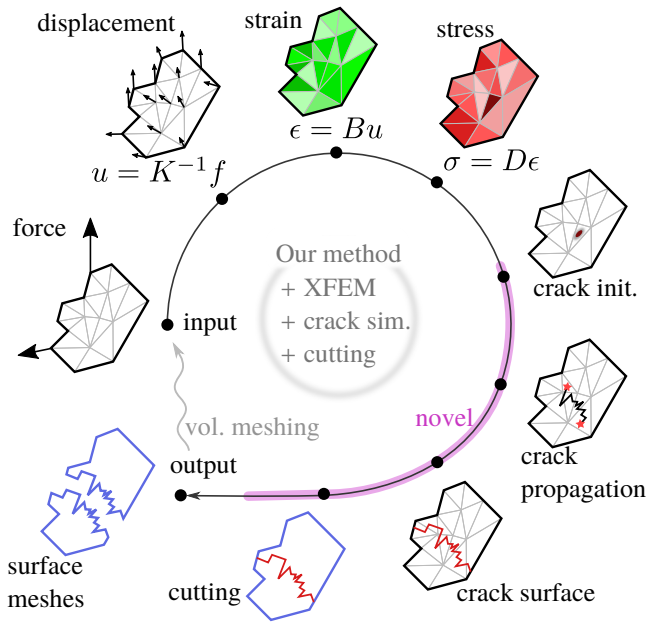
depending on whether the nodes are Heaviside enriched or crack-tip enriched respectively. The number of rows in the global, square system matrix  $\mathbf{K}$  increases from  $3n$  (in 3D) to  $3(n + n_s)$  in the case of Heaviside enrichment and to  $3(n + n_s + 4n_t)$  in the case of Heaviside and crack-tip enrichment. Here  $n$  is the number of nodes in the mesh,  $n_s$  is the number of nodes on elements with a complete crack passing through them and  $n_t$  is the number of nodes on elements containing crack front. In general,  $n_s + n_t \ll n$  since only the elements intersected by the crack surface are enriched.

#### 4. Method Overview

An overview of our method is illustrated in Fig. 3. Starting from a (detailed) surface mesh, we first convert it to a lower-resolution mesh to accelerate simulation and collision detection (see § 7), and then construct a tetrahedral mesh for XFEM.

With the tetrahedral mesh, we perform simulation by applying given boundary conditions and computing displacements and stress as in standard XFEM. We simulate crack growth by emulating the





**Figure 3:** Illustrative summary of the different stages of our method.

existence of singular “crack-tip stress”, correlating finite element displacements with crack-tip displacement equations in elements containing the crack front (§ 5.2). Thus, crack initiation and propagation can be handled separately (§ 5.3) as in linear elastic fracture mechanics (LEFM). Since we operate directly on an explicit surface mesh, our crack propagation algorithm proceeds as shown in Fig. 6, extending crack front vertices according to the strain energy release rate. We make use of an induced signed distance function by intersecting elements with our explicit crack surface mesh to evaluate enrichment functions—which we do after each propagation step.

Finally, we compute the disconnected mesh fragments using the generated explicit crack surface and the high detail mesh - for visualisation and further simulation (§ 6).

## 5. Displacement-Correlated XFEM

Assuming an unfractured object, boundary conditions, and a finite element mesh such that we can assemble global matrix  $\mathbf{K}$  and vector  $\mathbf{u}$ , we describe in § 5.1 how to add a crack to this system while using only Heaviside enrichment (Eq. (6)). The dynamics of fracture in a quasi-static and volumetric setting are then presented in § 5.2, which we use to calculate our fracture-mechanical loading parameters. We then describe our crack mesh representation and the steps to initiate a crack, and compute crack-front motion using the computed loading parameters in § 5.3.

### 5.1. System Equations

In XFEM, the crack is represented by two types of enrichments,  $\Upsilon_{\text{hev}}$  and  $\Upsilon_{\text{tip}}$ , which capture strong discontinuities in the displacement field and stress singularities respectively. Creating additional DOFs is the common property of these types of enrichments. However,

treating these enrichments simultaneously when formulating XFEM is challenging: In addition to being cumbersome to implement, the accuracy of crack-tip enrichment is limited in 3D as existing methods (which rely on 2D crack-tip parameters) require the stiffness matrix and force vector to be expanded arbitrarily to account for the extra unknowns. Accuracy further suffers from the lack of a clear definition of ‘normal to the crack front’ for points further away from the crack front. Another drawback is the introduction four times the DOFs compared to Heaviside enrichment (48 compared to 12) while performing extrinsic enrichment.

One possible way to circumvent this issue is *intrinsic* XFEM [FB06] which replaces the element shape functions by special ones with no additional unknowns to capture non-smooth solutions. This method, however, requires careful treatment of enriched moving least-squares functions near discontinuities as well as special weighting functions.

We propose a simpler *displacement correlated* XFEM on the basis of the extrinsic formulation but we reduce the system to only requiring Heaviside enrichment. We capture strong discontinuity as per Heaviside enrichment, but correlate analytical expressions for crack-tip displacement with numerically obtained smooth solutions to estimate stress intensities. A fracture can still be represented by a single sheet of triangles. Using this surface, the weak form of the linear elasticity equations (cf. Eq. 2.58 in [Kho15]) is applied after dropping the (zero) surface traction term. The reduced global system is

$$\begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{ua} \\ \mathbf{K}_{au} & \mathbf{K}_{aa} \end{bmatrix} \begin{Bmatrix} \mathbf{u} \\ \mathbf{a} \end{Bmatrix} = \begin{Bmatrix} \mathbf{f}_u \\ \mathbf{f}_a \end{Bmatrix}, \quad (9)$$

which we evaluate using standard quadrature, with subdivision to integrate enriched elements (we refer readers to Koschier *et al.* [KBT17] for a more accurate technique which does not require sub-division to integrate enriched elements). The equations for each block are given in the Appendix. Enrichment means that the blocks  $\mathbf{K}_{ua}$ ,  $\mathbf{K}_{au}$ , and  $\mathbf{K}_{aa}$  as well as the unknown vector  $\mathbf{a}$  will grow with each new element that is cut, but the size (and structure) of  $\mathbf{K}_{uu}$  and  $\mathbf{u}$  remains unchanged.

To resolve the issue of estimating stress intensities without crack-tip enrichment, we simply require that crack-front vertices have a reference element in which they reside (see § 5.2). Thus, there are no unknowns introduced at the crack front but we add DOFs according to the elements completely cut by the crack.

### 5.2. Fracture Dynamics

We now describe how finite element displacements (from Eq. (9)) near the crack front are used to emulate singular stress fields as in LEFM. This formulation allows us to propagate the crack using the strain energy release rate similarly to Hahn and Woltan [HW15], which is ideal for treating crack initiation and propagation separately (cf. § 5.3).

**Stress Intensities Near the Crack Front:** Using the displacement  $\mathbf{u}$  computed by solving Eq. (9), we can calculate stress and thus stress intensities near the crack front for propagation. We adopt the displacement method [CTW70] to estimate stress intensities by

correlating computed displacements with known crack-front displacement equations. These equations are evaluated at a location  $\mathbf{x}_\Gamma(r, \theta)$  which is on the crack face. This location is called the *correlation point*, where  $r$  and  $\theta$  are polar coordinates which are defined relative to the crack front (cf. Fig. 4).

Given  $\mathbf{x}_\Gamma(r, \theta)$ , the analytical displacement equations near the crack front are defined by

$$\begin{aligned} u_1^* &= \frac{K_I}{2\mu} \sqrt{\frac{r}{2\pi}} g_1^I(\theta) - \frac{K_{II}}{2\mu} \sqrt{\frac{r}{2\pi}} g_1^{II}(\theta) \\ u_2^* &= \frac{K_I}{2\mu} \sqrt{\frac{r}{2\pi}} g_2^I(\theta) + \frac{K_{II}}{2\mu} \sqrt{\frac{r}{2\pi}} g_2^{II}(\theta) \\ u_3^* &= \frac{2K_{III}}{\mu} \sqrt{\frac{r}{2\pi}} g_3^{III}(\theta), \end{aligned} \quad (10)$$

where  $u_{i=1,2,3}^*$  are the displacement components at  $\mathbf{x}_\Gamma(r, \theta)$ , and  $\mu$  is the shear modulus.  $K_{L=I,II,III}$  are the stress intensity factors (SIFs) which characterise the stress singularity - each corresponding to a crack loading mode shown in Fig. 5. Finally,  $g_i^L(\theta)$  are the angular functions (see Appendix), which describe the near-tip stress change along the circumference direction.

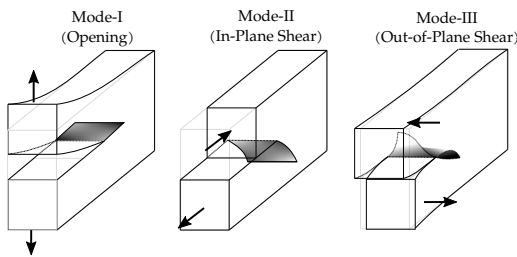
Eq. (10) can be further simplified from which corresponding expressions for  $K_{L=I,II,III}$  can be determined. This is because SIFs are evaluated on the crack face and relative to the crack front, which is the location of crack opening where  $\theta = \pi$  (cf. Fig. 4). Thus, for crack geometries with pure mode-I loading, crack front displacements reduce to

$$u_2^* = \frac{1}{2\mu} \sqrt{\frac{r}{2\pi}} K_I g_2^I(\theta). \quad (11)$$

The corresponding SIF is determined via conversion to give

$$K_I = 2\mu \sqrt{\frac{2\pi}{r}} \frac{u_2^*}{g_2^I(\theta)}, \quad (12)$$

which is the simplest method to determine  $K_I$ . Extending to pure



**Figure 5:** Arrows: Loading modes; Shaded: expected propagation behavior ([Irw57]). Mode-I opens the crack perpendicular to the crack plane and makes it propagate forward due to the applied tensile loading. In Mode-II, the crack faces are displaced on their plane, normal to the crack front. For Mode-III, the crack is displaced on its 'plane', parallel to the crack front.

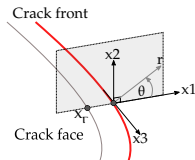
mode-II and mode-III loading for  $K_{II}$  and  $K_{III}$ , we have

$$[K_I \ K_{II} \ K_{III}] = \mu \sqrt{\frac{\pi}{2r}} \begin{bmatrix} \frac{u_2^*}{(2-2\nu)} & \frac{u_1^*}{(2-2\nu)} & u_3^* \end{bmatrix}, \quad (13)$$

which describes how we compute the SIFs that we use for crack propagation. The expressions for  $K_{II}$  and  $K_{III}$  in Eq. (13) are derived by applying similar interpretations as  $K_I$  but using  $g_1^{II}$  with  $u_1^*$ , and  $g_3^{III}$  with  $u_3^*$  in Eq. (11) and Eq. (12) respectively. The solution is simple, fast and sufficiently accurate for our purposes even though more accurate methods exist (e.g. J-integral method [IW03]).

**Virtual Crack Opening Displacements:** In practice, evaluating Eq. (13) is dependent on the relation between  $u_i^*$  and the solution  $\mathbf{u}$  (from Eq. (9)), which we now describe.

We can—for a moment—assume that the crack has already been initiated (as described in § 5.3), since Eq. (13) is given on the basis of the existence of a crack. Thus, let  $\mathbf{x}_\Gamma = \mathbf{x}_\Gamma(r, \theta)$ ; then, given  $\mathbf{x}_\Gamma$  we compute  $u_i^*$  from  $\mathbf{u}$  by exploiting polynomial approximations on the element  $e$  containing  $\mathbf{x}_\Gamma$  as interpolated nodal displacements  $\mathbf{u}_e(\mathbf{x}_\Gamma)$ , which we get using Eq. (2) after solving the linear system in Eq. (9). The location  $\mathbf{x}_\Gamma$  is given near the boundary of the crack mesh (e.g. a point along interior dashed blue line in Fig. 6, right). Finally, we project  $\mathbf{u}_e(\mathbf{x}_\Gamma)$  onto the local orthonormal basis at the crack-front (cf. Fig. 4). A disadvantage of this approach is the assumption that the interpolated displacements are representative of the crack opening displacements which leads to a loss of symmetry properties when all three modes I, II, and III are superimposed. However, we believe that the simplicity of this formulation outweighs the drawbacks since it results in fewer DOFs due to enrichment while still retaining the advantages of explicit XFEM.



**Figure 4:** Coordinate system perpendicular to crack front.

### 5.3. Crack Initiation and Propagation

Having described how we compute fracture loading parameters from finite element displacements, we now provide the details of how a crack is initiated, and propagated using the loading parameters from § 5.2.

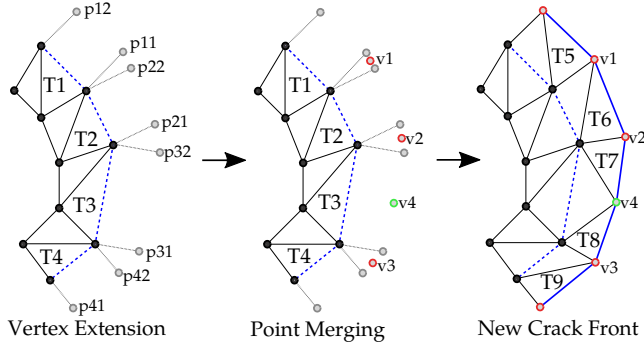
**Crack Mesh Initiation:** During XFEM, a crack is initiated according to a fracture criterion based on the Rankine condition - brittle material fails if the maximum principal stress exceeds material strength

$$\sigma_{\max} \geq \sigma_{\text{critical}}.$$

Crack initiation and its location are determined as follows: First, we calculate the nodal displacements followed by stresses in all elements. Then, eigen analysis is computed on each element's stress tensor to find direction of maximum tensile (or compressive) stress by comparing based on the largest magnitude. If the Rankine condition is satisfied, we then create an initial mesh at the element centre.

A 3D crack surface is represented by an explicit mesh which we initially create as a single sheet of triangles spanning a circular disk (other types of meshes are also possible e.g. rectangular mesh, which was in Fig. 10). We create and align this mesh to be orthogonal to the principal stress direction (tensile or compressive) such that it is

inline with expected crack propagation direction. Disk diameter is scaled according to element size, but may be set manually *e.g.* as a fraction of the domain's bounding box diagonal to control resolution. Since all the triangles of this disk mesh are vertex adjacent, each forms part of the initial crack front which will be propagated via their shared correlation point.



**Figure 6:** Crack mesh extension: Triangles T1-T4 form the current crack front (dashed blue line, left and middle). We extend two vertices for each triangle along the crack front, and then combine the extended points (red) or insert new ones *e.g.* v4 (green) according to the new edge length. Finally, we build triangles forming new crack front T5-T9 (solid blue line, right).

**Crack Mesh Propagation:** Propagation of the crack front occurs according to the strain energy release rate [And05] where the crack propagates iff  $K_* > K_t$ , where

$$K_* = \sqrt{\frac{K_I^2 + K_{II}^2 + K_{III}^2}{(1-\nu)}} \text{ and } K_t = \sqrt{\frac{2\gamma E}{(1-\nu^2)}}$$

are the effective stress intensity and material toughness.  $\gamma$  is the surface energy,  $E$  is Young's modulus and  $\nu$  is Poisson's ratio of an element incident to the crack-front. We calculate the propagation speed and direction at crack front vertices similarly to Hahn and Wolan [HW15] (see also [PM07]), but in a volumetric setting to propagate the fracture surface mesh. Propagation speed  $s = c_R(1 - K_*/K_t)$  is the linear approximation of the upper limit of the Rayleigh wave speed  $c_R \approx 0.57\sqrt{E/\rho}$  [Kun13], where  $\rho$  is the material density. We compute the propagation direction by

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

Starting from the crack front triangles, we extend the crack mesh during propagation as follows (*cf.* Fig. 6, left): Given a triangle, *e.g.* T2, we copy and extend its two vertices which belong to the current crack front - creating new temporary points *e.g.* p21 and p22. For convenience, vertices are extended on a triangle-by-triangle basis, so each crack front vertex will have two temporary copies which we later merge (see below). Crack front vertices are extended according to the propagation speed and direction.

To compute the connectivity defining the new crack front, we first merge temporary points, which ensures well-shaped triangles in the generated mesh. We merge pairs of temporary points that are copies

of the same vertex which is on the current crack front *e.g.* p11 and p22 (*cf.* Fig. 6, left). Merging produces new vertices like v1 which we compute simply as a mid-point (*cf.* Fig. 6, middle). After merging all paired points, we then add edges connecting adjacent mid-point vertices to form the new crack front geometry. Extra vertices may be added in this step to split the new edges if their length exceeds a given threshold, *e.g.* twice the crack front edge size in the initial disk mesh. Finally, we construct triangles to extend the crack surface by connecting the new crack front geometry (*cf.* Fig. 6, right, T5-T9).

## 6. Mesh Cutting

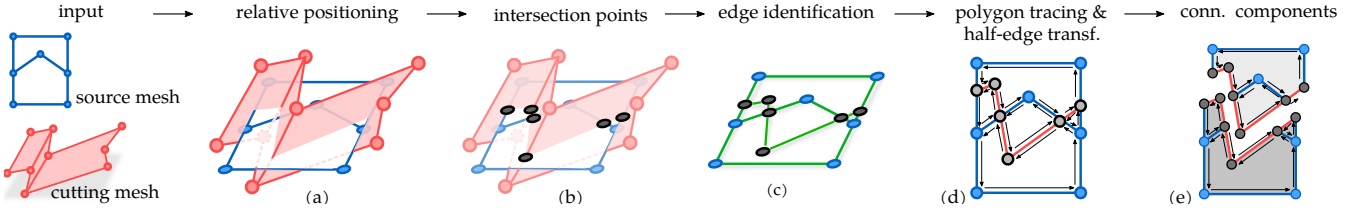
In this section, we describe our novel cutting algorithm that can be applied to arbitrary planar polygons for representing surface meshes to alleviate requirements for explicit triangulations or volumetric decompositions to cut domains. The algorithm is a practical solution to the problem of resolving complex intersections between meshes, which is a nuisance to implement. The method is particularly attractive as it also generalises to a number of topological settings (*e.g.* two-dimensional cuts and even boolean operations) enabling use-cases which are within—and even beyond—the scope of immediate application. In practice, we use this method to cut elements for enrichment, as well as cutting arbitrarily-shaped surface meshes which define the domain to obtain new fragments for simulation and rendering.

**Overview:** The input to our cutting algorithm is a pair of mesh data structures for an object  $\mathcal{M}$  and a crack surface  $\mathcal{C}$ . The output is a set of mutually exclusive fragment meshes which are a result of the cut. In our implementation, we make the simplifying assumptions that both meshes are composed of simple polygons (which can be concave); and that improbable cases like the intersection of two edges or a vertex intersecting a plane are extremely unlikely. An illustration of the pipeline in 2D is shown in Fig. 7: we use a 2D illustration since it is difficult to visualize the effect of operations on 3D meshes in a static figure.

Our algorithm can be implemented using any standard mesh data structure, (*e.g.* vertex-face adjacency list) but the halfedge mesh is most convenient since it supports maintaining incidence information of vertices, (half)edges and faces. For convenience, we resolve all intersections and connectivity in one polygon soup (*cf.* Fig. 7 (a)).

**Intersection Points:** We calculate intersection points by testing the halfedges of each polygon in  $\mathcal{M}$  against each polygon in  $\mathcal{C}$  using standard point-in-polygon tests [Hai94] (*cf.* Fig. 7 (b)). Our mesh vertex coordinates are assumed to be rational coordinates, thus intersection points are computed exactly, similarly to Zhou *et al.* [ZGZJ16]. In general, we impose no restrictions on the numerical representation of intersection points, so static filtered floating-point predicates can also be applied using the binary space-partition view as in [BF09], with some consideration for concave polygons (see also [She96]).

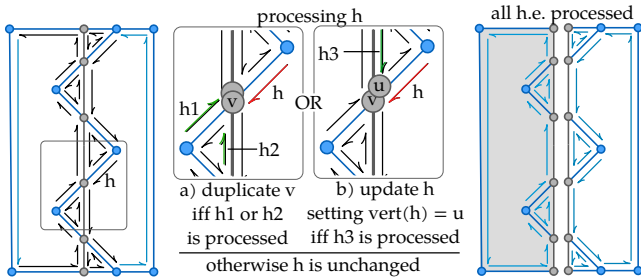
Once computed, we insert intersection points into the mesh data structure of the polygon soup and maintain a list of points that reside on each intersecting polygon. In practice, we speed up this process with a bounding volume hierarchy (BVH).



**Figure 7:** Our cutting algorithm consists of four different stages (b-e). We test all pairs of intersecting polygons (using a bounding volume hierarchy) and insert intersection points, as well as edges between these points, into a half-edge data structure. Then, we traverse all edges in the data structure, duplicate nodes when required and update edges to these nodes.

**Edge Identification:** We identify edges by connecting the intersection points of each pair of intersecting polygons between  $\mathcal{M}$  and  $\mathcal{C}$  (cf. Fig. 7 (c)). If exactly two intersection points result from a pair of intersecting polygons then we simply connect them, otherwise we sort and connect them according to the order of their coordinates. Since the intersection points are guaranteed to be collinear (the polygons are planar), we add edges consecutively in the order of the computed connectivity. We also identify edges between points which lie on the same edge from the original mesh of  $\mathcal{M}$  (and  $\mathcal{C}$ ) while ensuring that we create a minimal set of non-overlapping edges as described above.

**Polygon tracing and halfedge transformation:** Once new edges are created, we trace the boundaries of new polygons which are coincident to intersection points (cf. Fig. 7 (c)). The new edges are used to clip intersecting polygons: For each intersected polygon in  $\mathcal{M}$  and  $\mathcal{C}$ , we collect all edges whose defining vertices coincide on this polygon to trace halfedge loops which define the new polygons. Our approach is analogous to boundary tracking (e.g. the directed-body concept [IH92]) but we use only the gathered (half)edges and without numerical floating-point operations. Thus, our solution incorporates Sutherland-Hodgman [SH74] and Weiler-Atherton [WA77] polygon clipping under a single representation and in a three-dimensional setting.

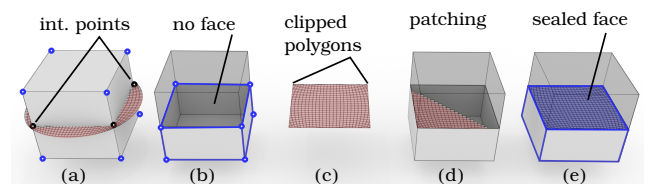


**Figure 8:** The intersection points (grey) due to a vertical cut (grey line) of two concave polygons (blue), are shown along with all half-edges (arrows). All new half-edges (black) are traversed sequentially to determine whether duplicate vertices are required.

Given  $\mathcal{M}$  and the new edges along its cut surface, we identify which intersection points need to be duplicated so that they appear on exactly two sides to partition resulting fragments (refer to Fig. 8). For this, we iterate through all half-edges  $h$  in  $\mathcal{M}$  which are from

the newly inserted edges, and process them individually using the following three conditions: 1) If  $h$  is incident on a vertex  $v$  and another halfedge on the same side as  $h$  (across the cut surface) and incident on  $v$  has been ‘processed’, then we update the connection of  $h$  to the correct instance of  $v$  so that it matches the other halfedge. 2) If this is not the case, we check if another halfedge connected to  $v$  on the opposite side of  $h$  (across the cut surface) has already been ‘processed’, in order to duplicate  $v$ . 3) If neither of the first two conditions are satisfied, then we leave  $h$  as is. Finally, we mark  $h$  as ‘processed’ and repeat the process on next halfedge of  $h$ . In practice, we traverse through halfedges one polygon at a time.

**Sealing:** Since the cutting is performed on a (hollow) surface mesh, the fragments of  $\mathcal{M}$  will not be closed just yet. For example, Fig. 9 (b) shows the bottom fragment of a cube cut by an elliptical mesh. The resulting fragment is a cuboid without the top face. To fix this, we construct polygon patches (similarly to Zhou et al. [ZGZJ16], and Mei and Tipper [MT13]) which are used to seal the fragments of  $\mathcal{M}$ . A patch is constructed as a set of adjacent polygons from  $\mathcal{C}$ , which is bounded by a closed sequence of halfedges forming an oriented loop and passing through intersection-points. We start from any new polygon in  $\mathcal{C}$  which is coincident to at-least two intersection points and use breadth-first search (BFS) to identify all patch-polygons. During BFS, we build a patch iteratively by adding adjacent polygons using a queue data structure. Two polygons are adjacent if one of them contains a halfedge whose opposite is in the other polygon. Once constructed, a patch is then duplicated to create a copy whose polygon winding order is reversed (e.g. clockwise). We then stitch each patch by matching its bounding halfedges with the border halfedges of each fragment of  $\mathcal{M}$  – inserting patch polygons one-at-a-time as shown in Fig. 9 (d). The final polygon-soup contains multiple mesh fragments as connected



**Figure 9:** Stitching the polygons of a cut-surface patch to a connected component of the input-mesh (cube).



components (cf. Fig. 7 (e)) which we determine using standard BFS routines.

## 7. Results and Discussion

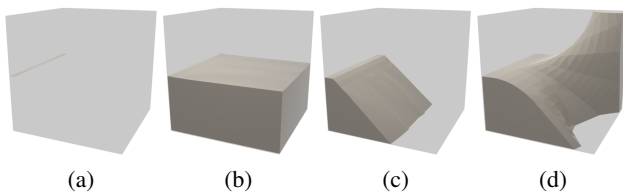
We present our results in this section. We show our fracture simulation results in § 7.1 and compare with related work (see accompanying video). Additional mesh cutting results are then shown in § 7.2.

**Implementation:** We visualize fragments using high detail but use lower resolution/simplified meshes to speed up simulation, tetrahedralization and mesh preprocessing for collision detection. The lower resolution meshes are created using standard edge-collapse [LT98] which is implemented with CGAL [Cac19]. Tetrahedralization is done with TetWild [HZG\*18] and the high-resolution fracture surfaces are simulated in the generated tetrahedral mesh. We enrich elements and cut the high-resolution meshes with our cutting algorithm which is implemented in C++ using the `Surface_mesh` data structure [SB12]. All results are rendered using Blender [Com18]. Rigid body dynamics are implemented with Bullet [Cou15], using VHACD [MG09] for convex decompositions to improve the robustness of collision detection.

Our Dirichlet boundary conditions are specified similarly to Müller *et al.* [MMDJ01]. Impact forces are computed at discrete collision events, where we treat objects as if they are anchored and proceed to compute their static equilibrium response using XFEM. We transfer impulses from Bullet similarly to Hahn and Woltan [HW16] (see also [GMD13]), based on the Hertz model and mapping collision points to the closest element in the XFEM mesh.

### 7.1. Fracture Simulation

We now show our fracture simulation results in this section. First, we show results for reproducing standard benchmarks and evaluate against expected crack propagation behaviour according to LEFM. We then show simple adaptations to emulate spatially varying material grain structure, which we use in our complex examples showing detailed fractures comprised of rigid-body animations.

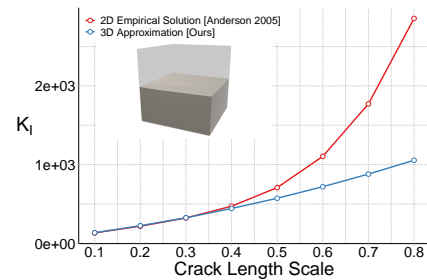


**Figure 10:** a) planar edge-crack under mode I, II, and III loading. Results: b) mode-I, c) mode-II, d) mode-III.

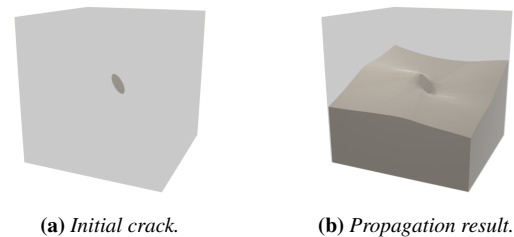
Fig. 10 shows propagation behaviour according to Irwin’s crack loading modes [Irw57] (see also Fig. 5). Our results are inline with theoretical predictions of LEFM by producing a crack which propagates according to the applied loading.

To assess the impact of our approximation quantitatively, we

compared stress intensity with an empirical equation for the single edge notch test in 2D (see Anderson [And05], Tab. 2.4). Our setup used a cube with edge length 4m consisting of 9987 elements (2048 nodes) and an initial crack surface with 60 faces. The material parameters used were: Loading=100N, Young’s modulus=2GPa, Poisson’s ratio=0.3, and density=2800 ( $\text{kg/m}^3$ ). Fig. 11 plots our estimate (blue) and the empirical solution (red) as the crack propagates (horizontal axis). Our method provides a linear approximation of a quadratic function, which is sufficient for low propagation steps but underestimates the SIF further away from the initial crack. For applications which do not require numerical fidelity, the advantage of our approximation is that it avoids crack-tip enrichment. Crack-tip enrichment, achieves better accuracy but at the cost of extra design complexity, increased computation and potential instability arising from the extra degrees of freedom.



**Figure 11:** Comparison: our approximation of SIFs vs empirical solution [And05] for the single edge-notched tension test (Mode-I).

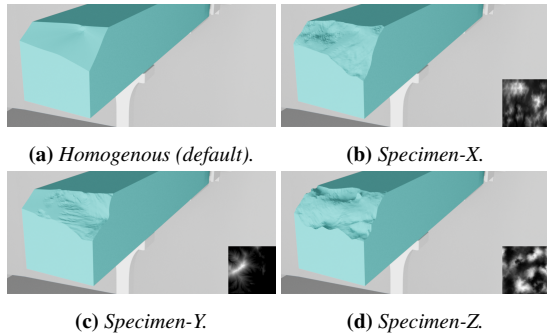


**Figure 12:** Crack propagation of the inclined crack under vertical loading

Fig. 12 shows our result for the “inclined penny crack” test, which is a crack propagation test often used in engineering to evaluate the validity of the simulation method. A circular surface crack is placed inside a cube with a  $45^\circ$  inclination relative to the axis of principal stress. Our method maintains qualitatively correct crack orientation in all cases. The fracture surface smoothly re-orient itself to be orthogonal to the direction of principal stress.

**Material Modulation:** Smooth surfaces in homogeneous material are important for brittle fracture simulation but they may lack the quality observed in real-world materials. Instead, we combine our already high-resolution cracks with surface-texture parametrization [DMA02] to obtain the distinctive quality sought in simple fashion. An example of our results is shown in Fig. 13. In addition, our method inherently supports spatial variations in the elasticity parameters enabling several avenues for biasing crack initiation as

shown in the example of Fig. 14. This is in contrast to methods such as BEM [HW15] where variations in the elasticity parameters require adjusting the fundamental solution.



**Figure 13:** Texture-based material grain structure which are used to emulate physically based toughness models. Here we show simple examples where a small fragment is chipped away from a block

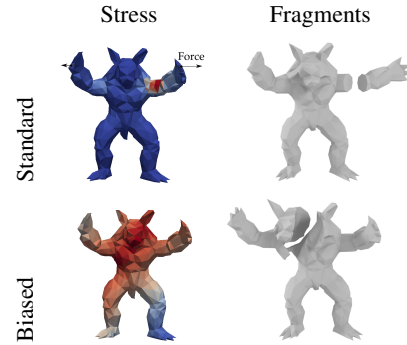
**Our Method vs. BEM [HW16]:** We performed a thin material breakage test where we compared our XFEM based method with the well known BEM. In the scenario shown in Fig. 16 we drop a wine glass on the floor. Because of the ground contact force, the glass breaks into multiple fragments. With our simulation (Fig. 16b), the entire glass shatters into multiple shards. In contrast, for the BEM simulation the bowl is noticeably unbroken resulting in a physically incorrect and even implausible state as cracks are under-resolved (Fig. 16a).

BEM can simulate impressive results in objects where the volume-to-surface-area ratio is sufficiently large. However, on objects with relatively larger surface area the method exhibits limitations in fracture behaviour on thin parts (and always dependent on the meshing details) with a number of causes including low-order integration, mesh resolution, and related numerical issues. Fractures tend to get ‘stuck’ and fail to cut off fragments properly (cf. Fig. 15). Also, we observed a particular susceptibility to a loss of intricate sharp features during cutting (cf. Fig. 17) due to a dependence on meshing operations with implicit surfaces which can be memory intensive to resolve.

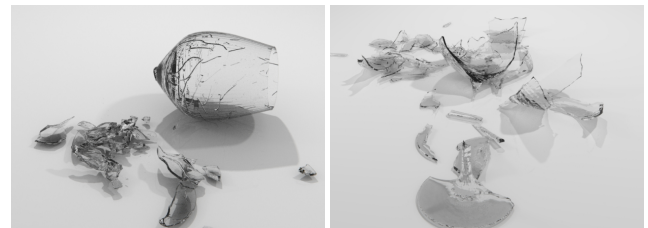


**Figure 15:** close-up model from Fig. 16a.

**Complex Brittle Fracture:** In addition to the comparisons and benchmark examples, we performed four simulations with multiple complex fractures. In the first experiment (cf. Fig. 18, left), the Stanford bunny is thrown at a wall. Although the object has complex concavities in its shape, the bunny is broken into many pieces. In the second example, we simulated a head-on impact collision between the Stanford armadillo and bunny (cf. Fig. 18, middle). Here we demonstrate that the rigid body coupling is able to handle fast paced interactions and/or robust collisions with many fragments. The third



**Figure 14:** Using material modulation to guide fracture. Considering homogeneous elasticity, high stress is within the inner left elbow (top-left) causing it to crack as shown in (top-right). By using a simple voxel embedding with stress-biasing coefficients, we weaken the middle of the armadillo relative to the rest of its body. This causes the highest stress to be in the chest area (bottom-left) and the crack is initiated this weak region (bottom-right).



(a) BEM

(b) XFEM (ours)

**Figure 16:** BEM [HW16] vs our method: Simulating brittle fracture in objects with thin features. Our method breaks the entire wine glass including the bowl and rim which are the thinnest parts unlike BEM which breaks only the stem and base.

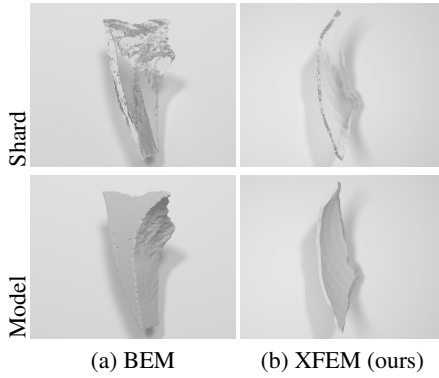
example (cf. Fig. 18, right) breaks a heavy marble statue fragmented into multiple pieces which retain their characteristic sharp features.

In the scenario illustrated in Fig. 1, we collide a wrecking ball with a statue which is then broken into many pieces. The result is another example of our ability to handle finely structured cuts with sharp edges on our fragments. Moreover, it shows that our method yields realistic results even on a relatively coarse mesh. The tetrahedral mesh is approximately 7k elements while the visualised/cutting mesh has 40k triangles.

Our performance results are shown in Table 1 which provides a breakdown of how long individual components took in each benchmark shown. These timings are based on our unoptimized and single-threaded implementation.

## 7.2. Polygon-Mesh Cutting

In this section, we show results for our mesh cutting algorithm. We first compare against similar mesh-based approaches and then show examples highlighting the generality of our approach.



**Figure 17:** Comparison of shard thickness and sharpness (cf. Fig. 16): The wine glass has thin parts which when broken produce shards with sharp edges. Here we show the difference between two example pieces from our simulation and BEM [HW16]. The piece from BEM is comes from the foot of the wine glass since it exhibits the finest breakage, whereas ours is from the bowl which is the thinnest part of the glass.

Example	$M_t$	$t_{cd}$	$t_{tet}$	$t_{sim}$	$t_{cut}$	$t_{tot}$
Fig. 1	40k	121	313	491	802	1539
Fig. 13	.3k	27	7	30	135	197
Fig. 16b	13k	1733	855	1411	329	3721
Fig. 18 (a)	8k	863	514	415	795	2471
Fig. 18 (b)	18k	788	529	203	365	1843
Fig. 18 (c)	16k	493	258	179	552	1437

**Table 1:** Performance overview: ( $M_t$ ) triangles defining input meshes (total); ( $t_{cd}$ ) convex decomposition time; ( $t_{tet}$ ) tetrahedralization time; ( $t_{sim}$ ) fracture simulation; ( $t_{cut}$ ) mesh cutting time; ( $t_{tot}$ ) total computation time of the entire simulation (includes I/O, mesh simplification etc.). Timings are given in seconds and measured on an Intel<sup>®</sup> Core<sup>™</sup> i9-7920X CPU @ 2.90GHz CPU.

**Our Method vs. Sifakis et al. [SDF07] and Wang et al. [WJST14]:** We compared our cutting results with state of the art [SDF07; WJST14], with respect to the number of triangles and connectivity. For this, we cut a Stanford bunny mesh using four different cutting surfaces (cf. Fig. 19). Table 2 summarizes the results. Compared to Sifakis et al., our method reduces the number of mesh edges and cutting elements (boundary facets) by 20% and 30% respectively. Our algorithm also produces fewer polygons and edges compared to Wang et al. [WJST14], and requires at least  $17\times$

Cutting Method	Repr.	#Vertices	#Edges	#SurfacePolys
Sifakis et al. [SDF07]	triangles	3420	10236	6824
Wang et al. [WJST14]	tetrahedra	25581	22300	84692
Ours	N-gons	3384	8226	4850

**Table 2:** Surface mesh cutting comparison: Using the scene shown in Fig. 19, we show the size (total) of mesh data (geometry and topology) which is to stored in order to cut the fragment meshes for each method.

fewer cutting elements (tetrahedra in their case) than theirs since we use a boundary representation of the cutting mesh. The visual quality of the meshes corresponding to the results in the Table 2 can be seen in Fig. 19.

**Concave Polygons and Polyhedra:** We demonstrate our cutting algorithm’s ability to cut concave polyhedra (cf. Fig. 20) and polygons by cutting a remodelled pentagonal frustum (blue) with a large quad (red). The pentagonal faces are modified so that there is a concavity, rotated so that they are not parallel to each other and divided into polygons with many concavities. The whole model is composed of only one volume element (all edges are on the surface). Our algorithm produces the correct surface meshes for the fragments (white), and does not modify the connectivity except where intersected with the cutting surface.

**General Examples:** In addition, we show our cutting algorithm’s ability to handle more general examples including partial cuts, 2D cuts and 3D boolean operations while still operating directly on the halfedge data structure. In Fig. 21a, we show a 3D partial cut where the input mesh is a cube with convex faces (no triangulation), and the cutting surface is composed of two triangles. Our algorithm correctly computes one output component with an incision-cut along three faces where the interior is sealed to form a water tight mesh (right). We show 2D cutting in Fig. 21b. In this example, our algorithm correctly cuts a surface mesh into two components with abutting convex and concave polygons. Finally, we show a generalisation to constructive solid geometry in Fig. 22, where we correctly handle a boolean operation between the Stanford bunny and a cube. Our algorithm produces the correct results for the classic set of operations (union, intersection and subtraction), and does not require additional information aside from the halfedge connectivity of the input surfaces.

## 8. Conclusion

XFEM is a simulation technique used in engineering which is usually tailored by domain experts for each application. We have presented an adaptation of XFEM applicable to brittle fracture simulation in computer graphics. Our algorithm improves the generality, efficiency, scalability and controllability of classical FEM by sacrificing accuracy. After introducing how we add cracks to the linear system, an approach to estimate stress intensity factors to compute crack propagation for XFEM was described. We demonstrated the advantages of using an explicit representation of the crack surface. Further, we showed that our method is able to realistically simulate detailed cracks – even with coarse tetrahedral discretizations. A general surface-cutting algorithm was also presented, which when combined with our explicit formulation enabled a decoupling of the resolution of the crack surface from the degrees of freedom. The cutting algorithm is exact in the sense that we only modify connectivity incident to the crack surface.

**Limitations and Future Work:** While we estimate the stress field within the volume, we perform cutting using surface meshes. The fragments therefore need to be tetrahedralized for recursive breakage to be possible. Although this introduces an extra computational cost,



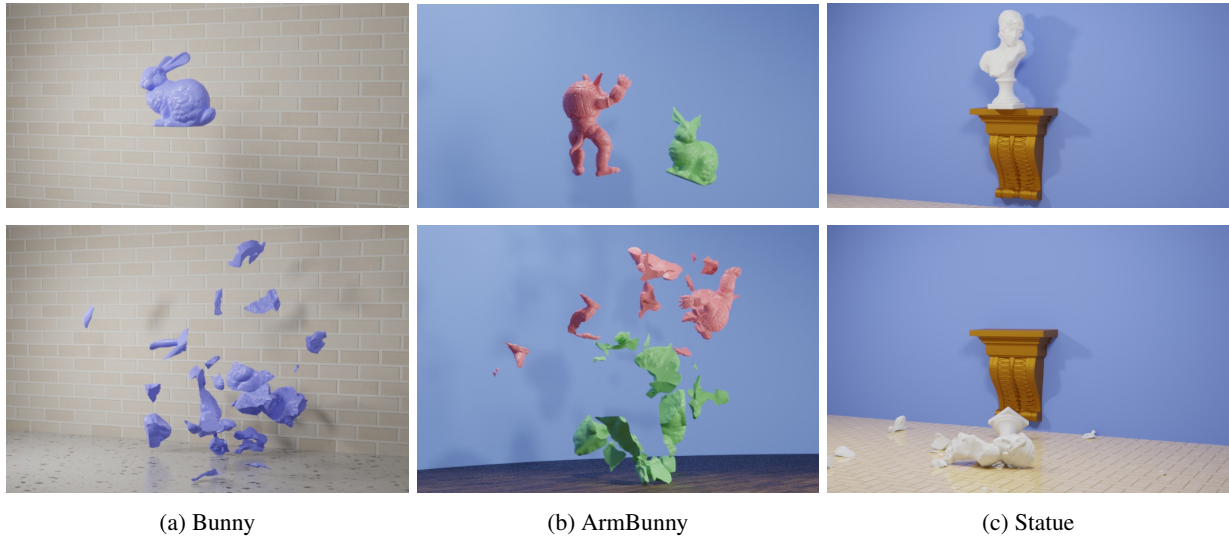


Figure 18: Examples of brittle fracture simulated with our method.

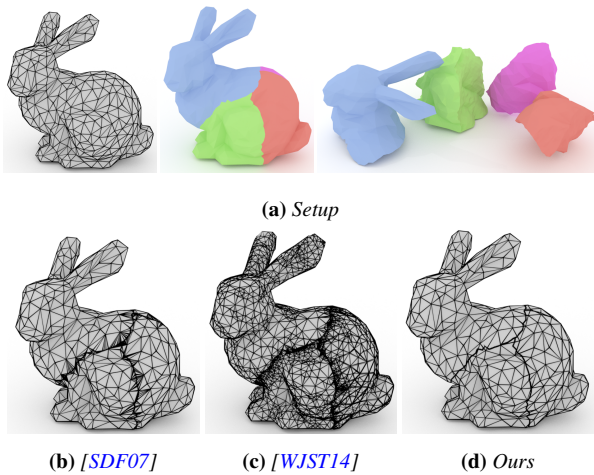


Figure 19: Mesh cutting evaluation scene where a bunny is cut into four pieces. See also Table 2.

our approach can deal with recursive breakage more efficiently than classical FEM, which requires the tetrahedral mesh to be refined during crack propagation. Our method only requires tetrahedralization once for each crack surface that generated fragments.

Another limitation of our method is that we assume simple topologies for the propagation of the crack front. We do not explicitly handle crack bifurcation (‘branching’ as in *e.g.* [DMD\*00; MGS11]), and self-intersections which can occur. Despite this, our method can still be used to simulate complex fracture patterns on flat (wine glass) and volumetric (statue) shapes.

Controlling the condition number of the stiffness matrix is an open challenge in XFEM discretization, which we did not address. Simulation failure can occur if the stiffness matrix is not kept regular, and/or when nodes whose enrichment functions have only

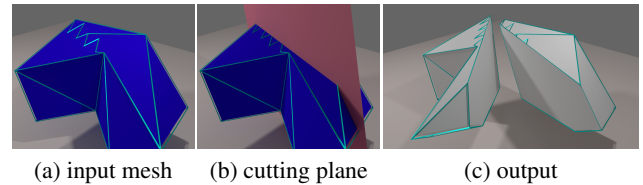


Figure 20: An extreme example. The input mesh (blue) is a pentagonal frustum where the pentagons (top and bottom faces) have been made concave (and are not parallel to each other). Each pentagon is composed of polygons with several concavities.

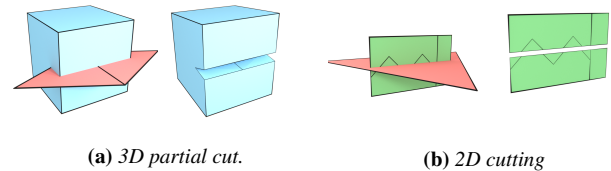


Figure 21: Examples of handling incision (partial) cuts and 2D cuts.

small supports in the cut element are not removed (see Fries and Belytschko [FB10] for a brief discussion).

Our cutting algorithm successively consigs the use of floating-point operations to the task of calculating intersection points but it is not provably robust against rare degeneracies. Despite this fact, the algorithm is reasonably stable, even with our complex fragment geometries. One solution to providing robustness guarantees is incorporating contingency measures to detect and resolve degeneracies by using tolerances in a hierarchical manner and specifying bounds on floating point error [WJST14]. Alternatively, geometric predicates may also be applied *e.g.* using the binary space-partition view for representing intersection points [BF09].



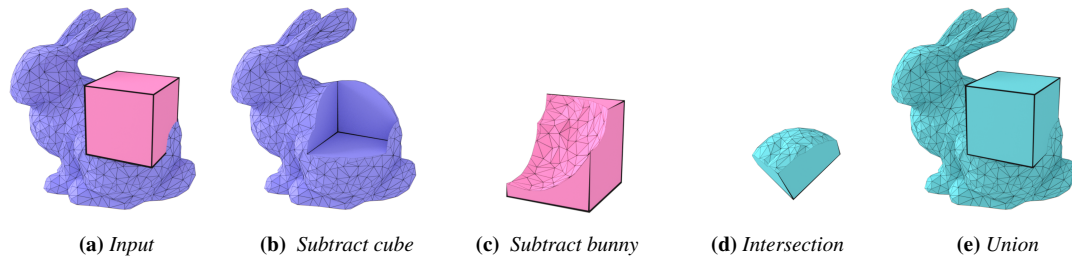


Figure 22: A boolean operation result using our mesh cutter

Finally, as mesh fragments often exhibit complex geometries, robust tetrahedral meshing tools are often required to mitigate unpredictable failures during tetrahedralization (and even simulation). Thus, the stability of simulation is also dependent on robust meshing tools (like TetWild [HZG\*18]) to cater for arbitrary shapes with intricate geometries.

## References

- [ADKK04] AOKI, K., DONG, N. H., KANEKO, T., and KURIYAMA, S. “Physically based simulation of cracks on drying 3D solids”. *Proceedings Computer Graphics International*, 2004. 2004, 357–364 2.
- [And05] ANDERSON, TED. *Fracture Mechanics: Fundamentals and Applications, Third Edition*. CRC Press, 2005. ISBN: 9781420058215 7, 9.
- [BB99] BELYTSCHKO, T. and BLACK, T. “Elastic crack growth in finite elements with minimal remeshing”. *International Journal for Numerical Methods in Engineering* 45.5 (1999), 601–620 3.
- [BF09] BERNSTEIN, GILBERT and FUSSELL, DON. “Fast, Exact, Linear Booleans”. *Proceedings of the Symposium on Geometry Processing*. SGP ’09. Berlin, Germany: Eurographics Association, 2009, 1269–1278 4, 7, 12.
- [BH07] BAO, ZHAOSHENG, HONG, JEONG-MO, TERAN, JOSEPH, and FEDKIW, RONALD. “Fracturing Rigid Materials”. *IEEE Transactions on Visualization and Computer Graphics* 13.2 (Mar. 2007), 370–378. ISSN: 1077-2626 2.
- [Cac19] CACCIOLA, FERNANDO. “Triangulated Surface Mesh Simplification”. *CGAL User and Reference Manual*. 4.13. CGAL Editorial Board, 2019 9.
- [Com18] COMMUNITY, BLENDER ONLINE. *Blender - a 3D modelling and rendering package*. Blender Foundation. Stichting Blender Foundation, Amsterdam, 2018 9.
- [Cou15] COUMANS, ERWIN. “Bullet Physics Simulation”. *ACM SIGGRAPH 2015 Courses*. SIGGRAPH ’15. New York, NY, USA: ACM, 2015. ISBN: 978-1-4503-3634-5 9.
- [CTW70] CHAN, S.K., TUBA, I.S., and WILSON, W.K. “On the finite element method in linear fracture mechanics”. *Engineering Fracture Mechanics* 2.1 (1970), 1–17. ISSN: 0013-7944 5.
- [CYFW14] CHEN, ZHILI, YAO, MIAOJUN, FENG, RENGUO, and WANG, HUAMIN. “Physics-inspired Adaptive Fracture Refinement”. *ACM Trans. Graph.* 33.4 (July 2014), 113:1–113:7. ISSN: 0730-0301 1, 2.
- [DBG14] DA, FANG, BATTY, CHRISTOPHER, and GRINSPUN, EITAN. “Multimaterial Mesh-based Surface Tracking”. *ACM Trans. Graph.* 33.4 (July 2014), 112:1–112:11. ISSN: 0730-0301 3.
- [DGW11] DICK, C., GEORGII, J., and WESTERMANN, R. “A Hexahedral Multigrid Approach for Simulating Cuts in Deformable Objects”. *IEEE Transactions on Visualization and Computer Graphics* 17.11 (Nov. 2011), 1663–1675. ISSN: 1077-2626 3.
- [DMA02] DESBRUN, MATHIEU, MEYER, MARK, and ALLIEZ, PIERRE. “Intrinsic Parameterizations of Surface Meshes”. *Computer Graphics Forum* 21.3 (2002), 209–218 9.
- [DMD\*00] DAUX, CHRISTOPHE, MOËS, NICOLAS, DOLBOW, JOHN, et al. “Arbitrary branched and intersecting cracks with the extended finite element method”. *International Journal for Numerical Methods in Engineering* 48.12 (2000), 1741–1760 2, 3, 12.
- [Do199] DOLBOW, JOHN EVERETT. “An extended finite element method with discontinuous enrichment for applied mechanics”. PhD thesis. Northwestern University, 1999 2, 3.
- [FB06] FRIES, THOMAS-PETER and BELYTSCHKO, TED. “The intrinsic XFEM: a method for arbitrary discontinuities without additional unknowns”. *International Journal for Numerical Methods in Engineering* 68.13 (2006), 1358–1385 5.
- [FB10] FRIES, THOMAS-PETER and BELYTSCHKO, TED. “The extended/generalized finite element method: An overview of the method and its applications”. *International Journal for Numerical Methods in Engineering* 84.3 (Oct. 2010), 253–304. ISSN: 1097-0207 12.
- [FB11] FRIES, THOMAS-PETER and BAYDOUN, MALAK. “Crack propagation with the extended finite element method and a hybrid explicit-implicit crack description”. *International Journal for Numerical Methods in Engineering* 89.12 (2011), 1527–1558 3.
- [GMD13] GLONDU, L., MARCHAL, M., and DUMONT, G. “Real-Time Simulation of Brittle Fracture Using Modal Analysis”. *IEEE Transactions on Visualization and Computer Graphics* 19.2 (2013), 201–209. ISSN: 1077-2626 2, 9.
- [GODB13] GARZON, J., O’HARA, P., DUARTE, C. A., and BUTTLAR, W. G. “Improvements of explicit crack surface representation and update within the generalized finite element method with application to three-dimensional crack coalescence”. *International Journal for Numerical Methods in Engineering* 97.4 (2013), 231–273 3.
- [Hai94] HAINES, ERIC. “Graphics Gems IV”. Ed. by HECKBERT, PAUL S. San Diego, CA, USA: Academic Press Professional, Inc., 1994. Chap. Point in Polygon Strategies, 24–46. ISBN: 0-12-336155-9 7.
- [HJST13] HEGEMANN, JAN, JIANG, CHENFANFU, SCHROEDER, CRAIG, and TERAN, JOSEPH M. “A Level Set Method for Ductile Fracture”. *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA ’13. New York, NY, USA: ACM, 2013, 193–201. ISBN: 978-1-4503-2132-7 2.
- [HW15] HAHN, DAVID and WOJTAN, CHRIS. “High-resolution Brittle Fracture Simulation with Boundary Elements”. *ACM Trans. Graph.* 34.4 (July 2015), 151:1–151:12. ISSN: 0730-0301 2, 3, 5, 7, 10.
- [HW16] HAHN, DAVID and WOJTAN, CHRIS. “Fast Approximations for Boundary Element Based Brittle Fracture Simulation”. *ACM Trans. Graph.* 35.4 (July 2016), 104:1–104:11. ISSN: 0730-0301 1, 2, 9–11.
- [HZG\*18] HU, YIXIN, ZHOU, QINGNAN, GAO, XIFENG, et al. “Tetrahedral Meshing in the Wild”. *ACM Trans. Graph.* 37.4 (July 2018), 60:1–60:14. ISSN: 0730-0301 9, 13.

- [IH92] IKEGAWA, Y. and HUDSON, J.A. “A Novel Automatic Identification System for Three-Dimensional Multi-Block Systems”. *Engineering Computations* 9.2 (1992), 169–179 8.
- [Irw57] IRWIN, GEORGE R. “Analysis of stresses and strains near the end of a crack traversing a plate”. *J. appl. Mech.* (1957) 6, 9.
- [IW03] INGRAFFEA, A.R. and WAWRZYNEK, P.A. “Finite Element Methods for Linear Elastic Fracture Mechanics”. *Comprehensive Structural Integrity*. Elsevier, 2003, 1–88. ISBN: 978-0-08-043749-1 6.
- [JK09] JEŘÁBKOVÁ, L. and KUHLÉN, T. “Stable Cutting of Deformable Objects in Virtual Environments Using XFEM”. *IEEE Computer Graphics and Applications* 29.2 (2009), 61–71. ISSN: 0272-1716 3.
- [JRC14] JUN, WU, RÜDIGER, WESTERMANN, and CHRISTIAN, DICK. “Real-Time Haptic Cutting of High-Resolution Soft Tissues”. *Studies in Health Technology and Informatics* (2014), 469–475. ISSN: 0926-9630 3.
- [KBT17] KOSCHIER, DAN, BENDER, JAN, and THUERREY, NILS. “Robust eXtended Finite Elements for Complex Cutting of Deformables”. *ACM Trans. Graph.* 36.4 (July 2017), 55:1–55:13. ISSN: 0730-0301 2, 5.
- [Kho15] KHOEI, AMIR R. *Extended finite element method: theory and applications*. Wiley series in computational mechanics. John Wiley & Sons, Ltd, 2015. ISBN: 978-1-118-45768-9 3–5, 15.
- [KLB14] KOSCHIER, DAN, LIPPONER, SEBASTIAN, and BENDER, JAN. “Adaptive Tetrahedral Meshes for Brittle Fracture Simulation”. *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '14. Eurographics Association, 2014, 57–66 2, 3.
- [KMB\*09] KAUFMANN, PETER, MARTIN, SEBASTIAN, BOTSCH, MARIO, et al. “Enrichment Textures for Detailed Cutting of Shells”. *ACM Trans. Graph.* 28.3 (July 2009), 50:1–50:10. ISSN: 0730-0301 2, 3.
- [KMBG08] KAUFMANN, PETER, MARTIN, SEBASTIAN, BOTSCH, MARIO, and GROSS, MARKUS. “Flexible Simulation of Deformable Models Using Discontinuous Galerkin FEM”. *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '08. Eurographics Association, 2008, 105–115. ISBN: 978-3-905674-10-1 2.
- [Kun13] KUNA, MEINHARD. *Finite Elements in Fracture Mechanics*. en. Vol. 201. Solid Mechanics and Its Applications. Dordrecht: Springer Netherlands, 2013. ISBN: 978-94-007-6679-2 978-94-007-6680-8 7, 15.
- [LT98] LINDSTROM, PETER and TURK, GREG. “Fast and Memory Efficient Polygonal Simplification”. *Proceedings of the Conference on Visualization '98*. VIS '98. IEEE Computer Society Press, 1998, 279–286. ISBN: 1-58113-106-2 9.
- [MBF05] MOLINO, NEIL, BAO, ZHAOSHENG, and FEDKIW, RON. “A Virtual Node Algorithm for Changing Mesh Topology During Simulation”. *ACM SIGGRAPH 2005 Courses*. SIGGRAPH '05. 2005 2, 3.
- [MCK13] MÜLLER, MATTHIAS, CHENTANEZ, NUTTAPONG, and KIM, TAE-YONG. “Real Time Dynamic Fracture with Volumetric Approximate Convex Decompositions”. *ACM Trans. Graph.* 32.4 (July 2013), 115:1–115:10. ISSN: 0730-0301 1.
- [MDB99] MOËS, NICOLAS, DOLBOW, JOHN, and BELYTSCHKO, TED. “A finite element method for crack growth without remeshing”. *International Journal for Numerical Methods in Engineering* 46.1 (1999), 131–150 2, 3.
- [MG04] MÜLLER, MATTHIAS and GROSS, MARKUS. “Interactive Virtual Materials”. *Proceedings of Graphics Interface 2004*. GI '04. 2004. ISBN: 1-56881-227-2 2.
- [MG09] MAMOU, KHALED and GHORBEL, FAOUZI. “A simple and efficient approach for 3D mesh approximate convex decomposition”. Nov. 2009, 3501–3504 9.
- [MGS11] MOUSAVI, S. E., GRINSPUN, E., and SUKUMAR, N. “Harmonic enrichment functions: A unified treatment of multiple, intersecting and branched cracks in the extended finite element method”. *International Journal for Numerical Methods in Engineering* 85.10 (2011), 1306–1322 3, 12.
- [MMDJ01] MÜLLER, MATTHIAS, MCMILLAN, LEONARD, DORSEY, JULIE, and JAGNOW, ROBERT. “Real-time Simulation of Deformation and Fracture of Stiff Materials”. *Proceedings of the Eurographic Workshop on Computer Animation and Simulation*. Manchester, UK: Springer-Verlag New York, Inc., 2001, 113–124. ISBN: 3-211-83711-6 2, 9.
- [MT13] MEI, GANG and TIPPER, JOHN C. “Simple and Robust Boolean Operations for Triangulated Surfaces”. *arXiv:1308.4434 [cs]* (Aug. 2013). arXiv: 1308.4434. (Visited on 04/30/2019) 8.
- [Mus13] MUSETH, KEN. “VDB: High-resolution Sparse Volumes with Dynamic Topology”. *ACM Trans. Graph.* 32.3 (July 2013), 27:1–27:22. ISSN: 0730-0301 3.
- [NTB\*91] NORTON, ALAN, TURK, GREG, BACON, BOB, et al. “Animation of fracture by physical modeling”. *The Visual Computer* 7.4 (1991), 210–219. ISSN: 1432-2315 2.
- [O'B02] O'BRIEN, JAMES F. “Graphical Modeling and Animation of Ductile Fracture”. *Proceedings of the 29th International Conference on Computer Graphics and Interactive Techniques. Electronic Art and Animation Catalog*. SIGGRAPH '02. ACM, 2002, 161–161. ISBN: 1-58113-522-X 2.
- [OH99] O'BRIEN, JAMES F. and HODGINS, JESSICA K. “Graphical Modeling and Animation of Brittle Fracture”. *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '99. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1999, 137–146. ISBN: 0-201-48560-5 2.
- [PDGJ09] PEREIRA, J. P., DUARTE, C. A., GUOY, D., and JIAO, X. “hp-Generalized FEM and crack surface representation for non-planar 3-D cracks”. *International Journal for Numerical Methods in Engineering* 77.5 (2009), 601–633 3.
- [PKA\*05] PAULY, MARK, KEISER, RICHARD, ADAMS, BART, et al. “Meshless animation of fracturing solids”. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2005)*. Vol. 24. 3. Association for Computing Machinery. Los Angeles: Association for Computing Machinery, 2005, 957–964 2.
- [PM07] PATRICIO, MIGUEL and MATTHEI, ROBERT M M. “Crack Propagation Analysis (CASA-report: Vol. 0723).” en. (2007), 25 7.
- [PNJO14] PFAFF, TOBIAS, NARAIN, RAHUL, de JOYA, JUAN MIGUEL, and O'BRIEN, JAMES F. “Adaptive Tearing and Cracking of Thin Sheets”. *ACM Trans. Graph.* 33.4 (July 2014), 110:1–110:9. ISSN: 0730-0301 3.
- [PO09] PARKER, ERIC G. and O'BRIEN, JAMES F. “Real-time Deformation and Fracture in a Game Environment”. *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '09. ACM, 2009, 165–175. ISBN: 978-1-60558-610-6 2.
- [PUC\*15] PAULUS, CHRISTOPH J., UNTEREINER, LIONEL, COURTECUISSIE, HADRIEN, et al. “Virtual Cutting of Deformable Objects Based on Efficient Topological Operations”. *Vis. Comput.* 31.6-8 (June 2015), 831–841. ISSN: 0178-2789 3.
- [RG17] REN, XIANG and GUAN, XUEFEI. “Three dimensional crack propagation through mesh-based explicit representation for arbitrarily shaped cracks using the extended finite element method”. *Engineering Fracture Mechanics* 177 (2017), 218–238. ISSN: 0013-7944 3.
- [RHS\*09] RICHARDSON, CASEY L, HEGEMANN, JAN, SIFAKIS, EFTYCHIOS, et al. “An XFEM method for modeling geometrically elaborate crack propagation in brittle materials”. *Institute for Computational and Applied Mathematics* 90095 (2009), 1555 2.
- [SB12] SIEGER, DANIEL and BOTSCH, MARIO. “Design, Implementation, and Evaluation of the Surface\_mesh Data Structure”. *Proceedings of the 20th International Meshing Roundtable*. Ed. by QUADROS, WILLIAM ROSHAN. Springer Berlin Heidelberg, 2012. ISBN: 978-3-642-24734-7 9.
- [SCM03] SUKUMAR, N., CHOPP, D.L., and MORAN, B. “Extended finite element method and fast marching method for three-dimensional fatigue crack propagation”. *Engineering Fracture Mechanics* 70.1 (2003), 29–48. ISSN: 0013-7944 3.
- [SCMB01] STOLARSKA, M., CHOPP, D. L., MOËS, N., and BELYTSCHKO, T. “Modelling crack growth by level sets in the extended finite element method”. *International Journal for Numerical Methods in Engineering* 51.8 (2001), 943–960 3.

- [SDF07] SIFAKIS, EFTYCHIOS, DER, KEVIN G., and FEDKIW, RONALD. "Arbitrary Cutting of Deformable Tetrahedralized Objects". *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '07. Eurographics Association, 2007, 73–80. ISBN: 978-1-59593-624-0 [2](#), [3](#), [11](#), [12](#).
- [SH74] SUTHERLAND, IVAN E. and HODGMAN, GARY W. "Reentrant Polygon Clipping". *Commun. ACM* 17.1 (Jan. 1974), 32–42. ISSN: 0001-0782 [8](#).
- [She96] SHEWCHUK, JOHNATHAN RICHARD. "Robust Adaptive Floating-point Geometric Predicates". *Proceedings of the Twelfth Annual Symposium on Computational Geometry*. SCG '96. ACM, 1996, 141–150. ISBN: 0-89791-804-5 [7](#).
- [SMMB00] SUKUMAR, N., MOËS, N., MORAN, B., and BELYTSCHKO, T. "Extended finite element method for three-dimensional crack modelling". *International Journal for Numerical Methods in Engineering* 48.11 (2000), 1549–1570 [3](#), [4](#).
- [SSC\*13] STOMAKHIN, ALEXEY, SCHROEDER, CRAIG, CHAI, LAWRENCE, et al. "A Material Point Method for Snow Simulation". *ACM Trans. Graph.* 32.4 (July 2013), 102:1–102:10. ISSN: 0730-0301 [2](#).
- [SSF09] SU, JONATHAN, SCHROEDER, CRAIG, and FEDKIW, RONALD. "Energy Stability and Fracture for Frame Rate Rigid Body Simulations". *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '09. ACM, 2009, 155–164. ISBN: 978-1-60558-610-6 [1](#).
- [SWB01] SMITH, JEFFREY, WITKIN, ANDREW, and BARAFF, DAVID. "Fast and Controllable Simulation of the Shattering of Brittle Objects". *Computer Graphics Forum* 20.2 (2001), 81–91 [2](#).
- [TF88] TERZOPOULOS, DEMETRI and FLEISCHER, KURT. "Modeling Inelastic Deformation: Viscoelasticity, Plasticity, Fracture". *SIGGRAPH Comput. Graph.* 22.4 (June 1988), 269–278. ISSN: 0097-8930 [2](#).
- [TSB\*05] TERAN, JOSEPH, SIFAKIS, EFTYCHIOS, BLEMKER, SILVIA S., et al. "Creating and Simulating Skeletal Muscle from the Visible Human Data Set". *IEEE Transactions on Visualization and Computer Graphics* 11.3 (May 2005), 317–328. ISSN: 1077-2626 [3](#).
- [WA77] WEILER, KEVIN and ATHERTON, PETER. "Hidden Surface Removal Using Polygon Area Sorting". *SIGGRAPH Comput. Graph.* 11.2 (July 1977), 214–222. ISSN: 0097-8930 [8](#).
- [WFL\*19] WOLPER, JOSHUAH, FANG, YU, LI, MINCHEN, et al. "CD-MPM: Continuum Damage Material Point Methods for Dynamic Fracture Animation". *ACM Trans. Graph.* 38.4 (July 2019), 119:1–119:15. ISSN: 0730-0301 [2](#), [3](#).
- [Wil61] WILLIAMS, M. L. "The Bending Stress Distribution at the Base of a Stationary Crack". *Journal of Applied Mechanics* 28.1 (Mar. 1961), 78–82. ISSN: 0021-8936 [4](#).
- [WJST14] WANG, YUTING, JIANG, CHENFANFU, SCHROEDER, CRAIG, and TERAN, JOSEPH. "An Adaptive Virtual Node Algorithm with Robust Mesh Cutting". *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '14. Eurographics Association, 2014, 77–85 [3](#), [4](#), [11](#), [12](#).
- [WMFB11] WOJTAN, CHRIS, MÜLLER-FISCHER, MATTHIAS, and BROCHU, TYSON. "Liquid Simulation with Mesh-based Surface Tracking". *ACM SIGGRAPH 2011 Courses*. SIGGRAPH '11. ACM, 2011, 8:1–8:84. ISBN: 978-1-4503-0967-7 [3](#).
- [WRK\*10] WICKE, MARTIN, RITCHIE, DANIEL, KLINGNER, BRYAN M., et al. "Dynamic Local Remeshing for Elastoplastic Simulation". *ACM SIGGRAPH 2010 Papers*. SIGGRAPH '10. 2010. ISBN: 978-1-4503-0210-4 [2](#).
- [WWD15] WU, JUN, WESTERMANN, RUDIGER, and DICK, CHRISTIAN. "A Survey of Physically Based Simulation of Cuts in Deformable Bodies". *Comput. Graph. Forum* 34.6 (Sept. 2015), 161–187. ISSN: 0167-7055 [4](#).
- [ZBG15] ZHU, YUFENG, BRIDSON, ROBERT, and GREIF, CHEN. "Simulating Rigid Body Fracture with Surface Meshes". *ACM Trans. Graph.* 34.4 (July 2015), 150:1–150:11. ISSN: 0730-0301 [2](#), [3](#).

- [ZGZJ16] ZHOU, QINGNAN, GRINSPUN, EITAN, ZORIN, DENIS, and JACOBSON, ALEC. "Mesh Arrangements for Solid Geometry". *ACM Trans. Graph.* 35.4 (July 2016), 39:1–39:15. ISSN: 0730-0301 [4](#), [7](#), [8](#).

## Appendix

**System Matrix Blocks:** The entries for the system matrix blocks in Eq. (9) are defined as follows (cf. Eq. 2.68 in [Kho15]):

$$\mathbf{K}_{uu} = \int_{\Omega} (\mathbf{B}^{\text{std}})^{\top} \mathbf{D} \mathbf{B}^{\text{std}}, \quad \mathbf{K}_{ua} = \int_{\Omega} (\mathbf{B}^{\text{std}})^{\top} \mathbf{D} \mathbf{B}^{\text{hev}} \quad (14)$$

$$\mathbf{K}_{au} = \int_{\Omega} (\mathbf{B}^{\text{hev}})^{\top} \mathbf{D} \mathbf{B}^{\text{std}}, \quad \mathbf{K}_{aa} = \int_{\Omega} (\mathbf{B}^{\text{hev}})^{\top} \mathbf{D} \mathbf{B}^{\text{hev}} \quad (15)$$

where,  $\mathbf{B}^{\text{std}} = \partial N / \partial \mathbf{x}$  and  $\mathbf{B}^{\text{hev}} = \partial N / \partial \mathbf{x} [N(\mathbf{x}) \mathbf{r}_{\text{hev}}(\mathbf{x})]$  are the gradient operators for the standard and enrichment shape functions respectively. The right hand side of Eq. (9) is assembled using the known element tractions  $\mathbf{t}$  as

$$\mathbf{f} = \begin{Bmatrix} \int_{\partial\Omega} (N^{\text{std}})^{\top} \mathbf{t} \\ \int_{\partial\Omega} (N^{\text{hev}})^{\top} \mathbf{t} \end{Bmatrix} \quad (16)$$

**Crack-tip Angular Functions:** The angular functions describing

near-tip stress change along the circumference direction in Eq. (9) are defined as follows (cf. Eq. 3.16, 3.25 and 3.31 in [Kun13]):

$$g_1^I = \cos \frac{\theta}{2} (\kappa - \cos \theta), \quad g_2^I = \sin \frac{\theta}{2} (\kappa - \cos \theta) \quad (17)$$

$$g_1^{II} = \sin \frac{\theta}{2} (\kappa + \cos \theta + 2), \quad g_2^{II} = -\cos \frac{\theta}{2} (\kappa + \cos \theta - 2) \quad (18)$$

$$g_3^{III} = \sin \frac{\theta}{2}, \quad (19)$$

where  $\kappa = 3 - 4\nu$  is the elastic constant for plane strain.