Data-Driven Approach for Simulating Brittle Fracture Surfaces

Yonghang Yu
The University of Tokyo
Tokyo, Japan
yonghangyu@graco.c.u-tokyo.ac.jp

Yuhang Huang
The University of Tokyo
Tokyo, Japan
yuhang.huang@graco.c.u-tokyo.ac.jp

Takashi Kanai
The University of Tokyo
Tokyo, Japan
kanai@graco.c.u-tokyo.ac.jp

ABSTRACT

In this paper, we propose a novel data-driven method that uses a machine learning scheme for formulating fracture simulation with the Boundary Element Method (BEM) as a regression problem. With this method, the crack-opening displacement (COD) of every correlation node is predicted at the next frame. In our naïve prediction, we design a feature vector directly exploiting stress intensities and toughness at the current frame, so that our method predicts the COD at the next frame more reliably. Thus, there is no need to solve the original linear BEM system to calculate displacements. This enables us to propagate crack-fronts using the estimated stress intensities. There are existing works which use the machine learning approach to accelerate the speed of traditional physics-based simulations like smoke and fluid, but our work is the first to incorporate the machine learning scheme into BEM-based fracture simulations. Our implementation accelerates the acquisition of displacements in linear time over the number of crack-fronts at each time step compared with the conventional solution whose time complexity grows exponentially based on the BEM linear system.

CCS CONCEPTS

• **Computing methodologies** → **Physical simulation**; **Classification and regression trees**; *Mesh geometry models*;

KEYWORDS

Brittle Fracture, Boundary Element Method, Data-Driven, Regression Forest

ACM Reference Format:
Yonghang Yu, Yuhang Huang, and Takashi Kanai. 2017. Data-Driven Approach for Simulating Brittle Fracture Surfaces. In *Proceedings of ACM SIGGRAPH ASIA 2017 Workshop: Data-Driven Animation Techniques (D2AT).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Rigid body fracture simulation is playing an increasinlgy important role in computer graphics applications such as games, movies, etc. A number of approaches for simulating the fracture of objects have been proposed in the past. Among them, one of the most used methods is the geometry-based method which employs pre-defined fracture patterns for the object when fractures are needed. However, this method is unable to carry out accurate physical computation.

D2AT, November 2017, Bankok, Thailand

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

Physics-based methods (including finite element method, mass-spring system, and meshless method) have excellent accuracy in expressing fracture patterns which reflect natural phenomenon better than geometry-based approaches. However, these methods cannot well describe details of fracture patterns like fracture surfaces. There are several methods for dealing with such issues such as re-meshing which can be used to increase details on fracture surfaces, but these methods are costly due to complex mesh manipulations.

Recently, physics-based fracture simulation by the boundary element method (BEM) is being proposed [Hahn and Wojtan 2015b]. This BEM method can well capture fracture surface details using a sheet of triangles. In addition, the number of elements for simulation can be reduced by using surface mesh instead of volumetric mesh. However, it is more inefficient to solve dense linear systems using the finite element method (FEM) compared to solving sparse linear systems with BEM. The costs of calculating displacements increase when the number of crack-fronts increases or when the fracture is complicated in BEM analysis. Although a fast approximation method to estimate displacements for BEM-based simulations was proposed by the same authors [Hahn and Wojtan 2016], the linear system still has to be solved for each frame of a simulation.

In this paper, we explore a totally different approach based on machine learning for the simulation of brittle fracture surfaces. There are existing works which use the machine learning approach to accelerate the speed of traditional physics-based simulations like smoke and fluid, but our work is the first to incorporate machine learning scheme into BEM-based fracture simulations. With our method, instead of solving the linear system for each frame of fracture surface simulation, the approach of learning and predicting is taken. In particular, we use a database to predict crack-opening displacements (COD), a parameter for solving linear equations. It enables us to approximate brittle fracture surfaces without solving linear equations.

To predict CODs, we adopt the regression forest [Breiman 2001] machine learning approach. With this approach, it is very important to determine the feature vector for ensuring good performance of regression forest, and consequently accurate prediction of CODs. Here we discuss feature vectors that can fully describe all the information needed to determine the next frame's COD, which will be used for computing stress intensities for propagation.

2 RELATED WORK

2.1 Geometry-based Fracture Animation

Terzopoulos and Fleischer [1988] first proposed fracture models in the computer graphics area. One approach for fracturing objects is the geometry-based method, also known as pre-defined model, which is a very popular method commonly used in computer games

or movies. Fracture patterns pre-defined by users are applied when the fracture phenomenon is needed or strong collision occurs. The generation of pre-defined fracture patterns requires tools that can control size as well as the shape of fractured fragments like Voronoi diagram. For example, Raghavachary [2002] describes a method of generating fracture fragments under the principle of Voronoi tessellation. Neff and Fiume [1999] propose a fracture model which allows rapid fracture by dividing a plane into fragments with the shape of polygon after the user specifies a specified angle. Mould [2005] also presents an image filter where the input is a drawing line constructed using Voronoi diagram. Based on this, the image of fractured surface is output. Although geometry-based approach allows users to control the fracture patterns, the fractures cannot reflect natural phenomenon dynamically. Also, the pre-defining step requires considerable preparation for designing fracture patterns. Müller et al. [2013] presents a method for resolving such issues by aligning a pre-defined fracture pattern with an impact location, allowing fast dynamic fracture of objects.

2.2 Fracture Simulation

Accurate simulation of fracturing objects is another approach of generating fractures in objects. The mass-spring system is one of the simplest method for simulating objects to be fractured. Within this system, the object is viewed as a set of particles with mass and position connected by springs pairwisely. Norton et al. [1991] modeled objects from simple lattice cubic cells, then generated fractures by breaking the link between cubes. Hirota et al. [1998] used the mass-spring model to generate crack fracture patterns like drying surface layer for many objects like surface of roads and drying mud. Mazarak et al. [1999] simulated explosions using connected voxel representation of objects which make the explosions more realistic by replacing flat artificial slices with more natural volumetric segments. Mass-spring models are sometimes popular due to their simplicity in modeling, but it is hard to describe physical quantities like strain and stress using the mass-spring system. Also, the orientation of the fracture plane cannot be defined.

Finite element analysis can well depict stress and strain relationships. O'Brien and Hodgins [1999] proposed brittle fracture simulations with the finite element method (FEM). Later, O'Brien et al. [2002] again presented ductile fracture as well. Although FEM can well define the oreintation of fractures, it cannot determine the forward direction of crack tips. Besides, this method also suffers from artifacts due to stress-based fracture criteria. On the other hand, Müller et al. [2001] proposed a hybrid method where static analysis is independent of time-step allowing simulating of fractures of objects in real-time. Bao et al. [2007] treated material as fully rigid body to solve issues introduced by small time step restrictions. Glondu et al. [2013] used modal analysis to initiate and propagate cracks with energy based algorithms. Their approaches are based on non-linear FEM analysis where the object is represented as discretized tetrahedral meshes.

The boundary element analysis of fractures by representing objects as triangle meshes has been proposed recently. Rather than simulating fractures with finite elements, our research is based on the boundary element method developed by Hahn and Wojtan [2015b] where displacements are computed directly by solving the resulting dense BEM system. Later, they proposed fast approximations of displacements and stress intensities for simulating BEM-based fractures [Hahn and Wojtan 2016], but a linear system still has to be solved. Zhu et al. [2015] presented fractures with surface meshes by solving the layer potential, after which stress analysis is performed with displacements by integrating potentials.

2.3 Data-Driven Approach for Simulations

Various types of data-driven based approaches for speeding up fluid simulations have been explored. A data-driven based approach that formulates fluid simulation as a regression problem with regression forest has been proposed, where the acceleration of a particle for each frame is predicted using a trained regressor [Ladický et al. 2015]. The projection step is a time-consuming step in grid-based fluid simulation. Yang et al. [2016] proposed a data-driven approach with artificial neural network (ANN) to migrate iterative computational costs, where the results of the projection step can be obtained in constant time for each grid cell. The operator splitting method within standard fluid solvers has to solve ill-conditioned linear equations, and the Convolutional Network (ConvNet) serves as a subsitute for realizing fast and realistic simulations [Tompson et al. 2016].

These researches aim to accelerate fluid simulation with data-driven based approaches. However, there are very few researches presented for brittle fracture simulation. By combining learning methods with physical brittle fracture simulation, our research has obtained fast and highly similar results as traditional solvers.

3 SIMULATING BRITTLE FRACTURES WITH BOUNDARY ELEMENTS

This section briefly reviews brittle fracture simulation with boundary elements from the previous works of Hahn and Wojtan [2015b]. We describe their work especially from the view of constructing brittle fracture surfaces including crack initialization and crack propagation.

Starting from a given highly detailed surface triangle mesh, it is transformed to a coarse mesh (called BEM mesh), which is used in BEM linear system computation. Known boundary conditions are then applied to a BEM mesh, and an initial BEM linear system in Equation (1) is then obtained [Kielhorn 2009]. Surface stress with displacements can be computed, and the new cracks will be initiated if the element's principal stress is larger than material strength,

$$\begin{bmatrix} \mathbf{V} & -\mathbf{K} \\ \mathbf{K}^T & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ \mathbf{u} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_D \\ \mathbf{f}_N \end{bmatrix}, \tag{1}$$

where $\mathbf{u}$ and $\mathbf{q}$ refer to displacements of boundary and tractions, which are unknown variables, and the known coefficient matrices including matrix blocks $\mathbf{V}, \mathbf{K}, \mathbf{D}$ are determined by initial mesh structures. In particular, $\mathbf{V}$ is a symmetric positive definite matrix. The right hand side of the linear system is known boundary data including the Dirichlet and Neumann boundary that specifies the boundary values along the boundary of the computational domain. They use Schur complements to solve this system (see Equation (5.18) in [Kielhorn 2009]).

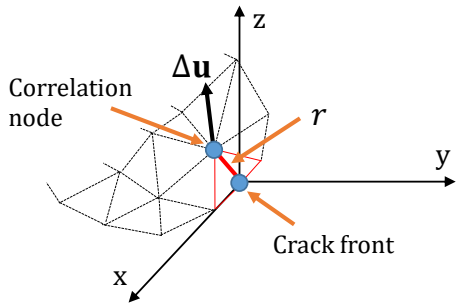

Figure 1: Displacement correlation technique. $\Delta\mathbf{u}$ is evaluated on correlation node of crack-front.

A new linear system is launched where the new cracks are added during crack propagation in Equation (2). This is from the formulation of Symmetric Galerkin Boundary Element Method (SGBEM) in [Frangi et al. 2002], which solves for crack-opening displacements (CODs) $\Delta\mathbf{u}$, norm of which is the distance between two faces with opposite surface normals of the crack,

$$\begin{bmatrix} \mathbf{V} & -\mathbf{K} & -\mathbf{K}_c \\ \mathbf{K}^T & \mathbf{D} & \mathbf{D}_c \\ \mathbf{K}_c^T & \mathbf{D}_c^T & \mathbf{D}_{cc} \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ \mathbf{u} \\ \Delta\mathbf{u} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_D \\ \mathbf{f}_N \\ 0 \end{bmatrix}. \tag{2}$$

The sizes of matrix blocks $\mathbf{K}_c, \mathbf{D}_c, \mathbf{D}_{cc}$ and $\Delta\mathbf{u}$ become larger accordingly when new cracks are added to the BEM mesh, whereas the matrix elements $\mathbf{V}, \mathbf{K}, \mathbf{D}$ computed beforehand remain unchanged.

Crack-opening displacements $\Delta\mathbf{u}$ are obtained by solving Equation (2), and then stress intensities are calculated by the displacement correlation technique from [Ingraffea and Wawrzynek 2003],

$$\begin{aligned} K_{\mathrm{I}} &= \mu\sqrt{2\pi}\frac{\Delta\mathbf{u}_{\mathrm{I}}}{\sqrt{r}(2-2\nu)}, \\ K_{\mathrm{II}} &= \mu\sqrt{2\pi}\frac{\Delta\mathbf{u}_{\mathrm{II}}}{\sqrt{r}(2-2\nu)}, \\ K_{\mathrm{III}} &= \mu\sqrt{\pi}\frac{\Delta\mathbf{u}_{\mathrm{III}}}{\sqrt{2r}}, \end{aligned} \tag{3}$$

where ν is a Poisson's ratio, μ is a Lamé parameter, and r is the distance from a correlation point to its corresponding crack-front as illustrated in Figure 1. $\Delta\mathbf{u}$ is evaluated at the correlation node which is the interior node of a triangle that contains the crack-front edge. Then $\Delta\mathbf{u}$ is projected onto a local coordinate system (x, y, z) of the crack-front, thereby resulting in the acquisition of $\Delta\mathbf{u}_{\mathrm{I}}$, $\Delta\mathbf{u}_{\mathrm{II}}$, and $\Delta\mathbf{u}_{\mathrm{III}}$.

Effective stress intensities are then calculated as ${K_{\mathrm{eff}}}^2 = {K_{\mathrm{I}}}^2 + {K_{\mathrm{II}}}^2 + {K_{\mathrm{III}}}^2/(1-\nu)$, to determine whether the cracks should propagate. Material toughness is defined as ${K_c}^2 = 2\gamma E/(1-\nu^2)$, where E is Young's modulus and γ is the material surface energy (see paper [Freund 1998] for details). Cracks propagate if $K_{\mathrm{eff}} \geq K_c$. The detailed crack propagation including propagation speed and direction is described in [Hahn and Wojtan 2015b].

4 DATA-DRIVEN SIMULATION OF FRACTURE SURFACES

The BEM-based brittle fracture simulation summarized in Section 3 is physically accurate. However, the cost becomes very expensive when solving the BEM linear system for crack-opening displacements (CODs) in each step of the simulation with the number of crack-fronts increasing. This section introduces prediction-based approximation of crack-opening displacement instead of solving linear equations.

4.1 Predicting Crack-Opening Displacements with Regression Forest

Previously, fracture simulations using the BEM solver were performed by solving linear equations at each step. However, it is very costly to solve linear equations every time the COD is changed. In addition, the time cost also becomes larger accordingly when CODs get larger. $\mathbf{q}$ and $\mathbf{u}$ are changed at each frame in Equation (2), but they are not used for crack propagation. On the other hand, $\Delta\mathbf{u}$ is necessary for obtaining stress intensities during crack propagation. We thus only estimate CODs through our prediction method, where we regard $\mathbf{q}$ and $\mathbf{u}$ as unchanged value and zero respectively.

To predict CODs, we adopted the machine-learning approach using regression forest [Breiman 2001]. Reasons why the use of regression forest is appropriate are; (1) it can handle continuous values for predictions such as CODs, (2) it can fit large-scale data, and (3) both training and prediction can be carried out quickly.

Feature vector design. What is to be addressed here is how to design the feature vectors for regression forests. By calculating stress intensities through crack-opening displacements in Section 3, we know they have strong relevance with CODs where CODs need to be exploited to calculate stress intensities. Thus we consider using stress intensities as part of our feature vector, where stress intensities at the current frame determine CODs at the next frame. Our feature vector should well describe the factors influencing the jump of two fracture surfaces. Three candidate feature vectors were tested in our research, namely $(K_{\mathrm{I}}, K_{\mathrm{II}}, K_{\mathrm{III}})$, $(K_{\mathrm{I}}, K_{\mathrm{II}}, K_{\mathrm{III}}, K_c)$ and $(K_{\mathrm{I}}, K_{\mathrm{II}}, K_{\mathrm{III}}, K_c, r)$ respectively. K_c is the toughness set for material, which is constant of some models, or changeable for some other models. r is distance for which the crack-front has propagated. The toughness also affects the CODs of the next frame if it is a variable. Larger CODs are obtained with smaller toughness. We chose the most proper feature vector $(K_{\mathrm{I}}, K_{\mathrm{II}}, K_{\mathrm{III}}, K_c)$ from the above three candidates, and conducted experiments described later.

The restriction for fast regression methods is the need to calculate feature vector constantly. Stress intensities, toughness, and distance can all be calculated in constant time. This is the linear-time calculation taken according to the number of crack fronts for each frame when estimating CODs.

4.2 Overview of Our Method

Figure 2 shows an overview of our method. In this figure, we list both the BEM-based fracture simulation system and our data-driven based fracture simulation process respectively. Our method aims to replace the original BEM solver with our regression forest based prediction. We only perform physical computation with the BEM solver at the first frame, and then prediction from the second frame.

We propose a novel data-driven method that uses a machine learning scheme which predicts projections $\Delta\mathbf{u}_{\mathrm{I}}, \Delta\mathbf{u}_{\mathrm{II}}, \Delta\mathbf{u}_{\mathrm{III}}$ of the correlating node of the crack-opening displacement of the next

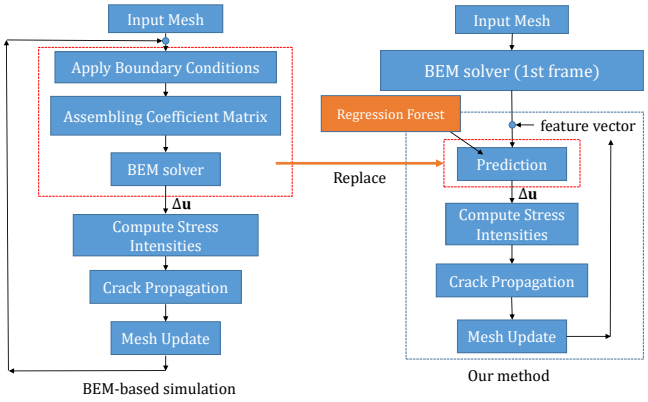

Figure 2: Left: Flowchart of BEM-based fracture simulation system. Right: Flowchart of our prediction system.

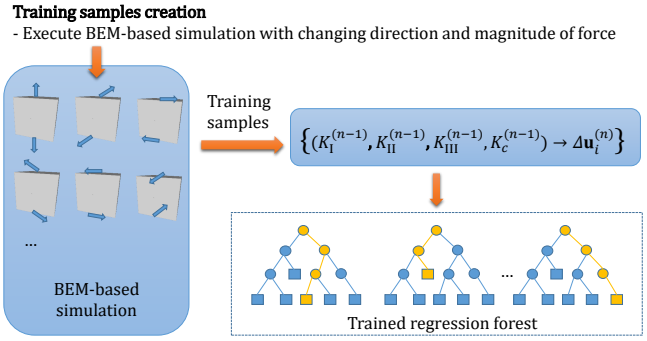

Figure 3: Creation of training samples and the construction of regression forest.

frame onto the local coordinate system of the current frame's crack-front. We designed a feature vector with stress intensities $K_{\mathrm{I}}, K_{\mathrm{II}}, K_{\mathrm{III}}$ and material toughness K_c of the current frame's crack-front containing information on determining the next frame's crack-opening displacements of correlating nodes.

However, we do not predict crack-opening displacement $\Delta\mathbf{u}$ directly in our implementation. Instead, we use its projections $\Delta\mathbf{u}_{\mathrm{I}}$, $\Delta\mathbf{u}_{\mathrm{II}}$, $\Delta\mathbf{u}_{\mathrm{III}}$ onto the local orthonormal coordinate system of the current frame's crack-front as our label. Crack-opening displacement (COD) is defined in the world coordinate system, however, stress intensities are calculated in the local coordinate system of the crack-front. The direct use of CODs as our label is not proper since the CODs in the world coordinate system are rotation-variant values. We thus use their projections onto the local coordinate system for crack-fronts, to ensure that our naïve prediction is performed at the same coordinate system.

With predicted projections of CODs, we then convert projections to the COD of the next frame through the transformation from the local coordinate system to the world coordinate system. Approximated CODs of correlating nodes at the next frame are then assigned to corresponding crack-fronts. Stress intensities can then be acquired through the displacement correlation technique in Section 3. The crack propagation proceeds in the same way as the previous BEM-based fracture simulation of [Hahn and Wojtan 2015b].

4.3 Training Samples Creation and Learning Stage

Figure 3 shows the creation process for training samples. We usually create a large number of training samples by executing BEM-based simulation many times where we use random force directions with different magnitudes as our initial conditions for BEM-based fracture. For each direction with a force magnitude, we obtain a set of training samples under this condition. Our training for regression forest is done with the method as in [Breiman 2001] by creating a set of decision trees learned for each subset in the training stage and by averaging all predictions from individual trees as final prediction. At the training stage, subsets from the original dataset are constructed with the bootstrap aggregating approach, where each training sample of the subset can be chosen randomly with replacement.

A training sample is defined as,

$$\left\{\left(K_{\mathrm{I}}^{(n-1)}, K_{\mathrm{II}}^{(n-1)}, K_{\mathrm{III}}^{(n-1)}, K_c^{(n-1)}\right) \rightarrow \Delta\mathbf{u}_i^{(n)}\right\}, \tag{4}$$

where $K_{\mathrm{I}}^{(n-1)}, K_{\mathrm{II}}^{(n-1)}, K_{\mathrm{III}}^{(n-1)}, K_c^{(n-1)}$ refers to stress intensities and material toughness of the crack-front at current frame $n-1$, where $\Delta\mathbf{u}_i^{(n)}$ refers to the projections of the COD of the interior node at the next frame n, and $i = \mathrm{I}, \mathrm{II}$ or III, which are three axes to which COD is projected onto.

In our implementation, we do not directly predict three projections with only one regressor, instead we split our training data into three parts, where each part only contains one projection as the label. Thus we have three regressors which are used to predict anyone of the three projections when inputting the same feature vector.

5 RESULTS AND DISCUSSION

We perform experiments on a desktop PC with Intel® Core™ i7-2600 3.40GHz CPU and 32GB RAM. We evaluate our method mainly by comparing our data-driven based results with BEM-based fracture simulation results. We have three different databases including Cube database, Armadillo database and Bar database. We will describe how to construct them later. For BEM-based brittle fracture simulation, we use FractureBEM source code [Hahn and Wojtan 2015a] provided by the authors of [Hahn and Wojtan 2015b] and HyeNA library [Graz University of Technology 2016] provided by the Institute of Applied Mechanics, Graz University of Technology. In the training and prediction stage, we also use the source code [Perception and Robotics Group 2014] provided by Machine Perception and Robotics Group, Chubu University as our regression forest implementation. We only consider two important parameter settings, which are the number of trees t and maximal depth of the tree d in the training stage. In later discussions, we denote such parameters as (t, d).

Prediction of Cube's fracture. As discussed in Section 4.3, the training samples are created by changing the direction and magnitude of force applied to object. For directions and magnitudes

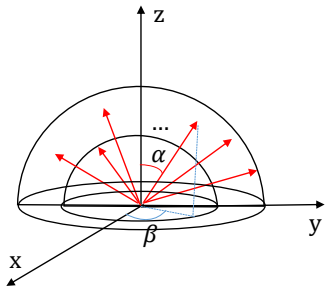

Figure 4: Random force creation. α is the slope angle and β is the rotation angle.

of forces applied to a Cube model, we here created any direction which is sampled randomly within the range of hemisphere and any magnitude which is between two different hemispheres' radii.

Figure 4 shows how to generate different forces randomly. This figure only illustrates one side's force, there is also another force which has the same pull intensity but with different direction in another side, both of which are applied to cube at the same time. For each force generated by our random process, we obtain a set of training samples through a BEM-based simulation. For Cube database, we have 1,235,511 training samples. The force is applied on the surface's same location every time.

We first test Cube database with our prediction method on a Cube triangle mesh by applying forces with the same magnitude but different directions whose $\alpha = 0$, 45 and 90 degree and $\beta = 0$ degree as our initial condition, where α is the slope angle and β is the rotation angle as illustrated in Figure 4. Parameters including the number of trees and the maximal depth of regression forest are (50,10) for the Cube database in the training stage.

The results of our method compared with the BEM-based simulation are shown in Figure 5. In this figure, fracture patterns at the last frame for different directions of initial forces applied to Cube are shown. In the first two columns of the figure, one half of the split Cube is shown. In the third column, un-fractured Cube is shown whose inner shaded surface is the fracture surface. It can be seen that our method can well approximate the BEM-based simulation results.

Figure 6 shows logarithmic graphs of 0 and 45 degree's stress intensities by our method and BEM-based simulation. In these graphs, "K_{I} BEM", "K_{II} BEM", and "K_{III} BEM" refer to stress intensities obtained by the BEM-based fracture simulation. On the other hand, "K_{I} multi D", "K_{II} multi D", and "K_{III} multi D" represent stress intensities obtained by our prediction method with random database. The results show that our method can obtain good estimates for stress intensities. Although the difference of K_{III} between our method and the BEM-based fracture simulation seems larger than that of K_{I} and K_{II}, K_{III} is a smaller scaled value compared with K_{I} and K_{II}, thus we still consider that the difference is not so large.

Other models. We also evaluate our method on an Armadillo model and a Bar model with their respective databases.

We create databases for the Armadillo and Bar by changing only the force magnitudes for the same direction. For the Armadillo's database, we have 150 different pull intensities for the horizontal

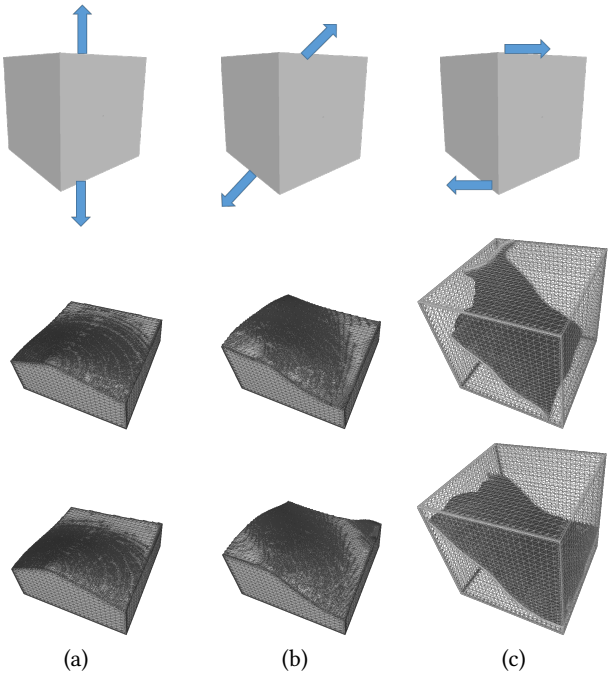

Figure 5: Initial forces applied on Cube model are shown in the first row. The second row refers to BEM-based simulation results, and the top bottom row is our data-driven results. From left to right are 0,45, and 90 degree test cases respectively.

direction. The number of sample nodes is 12,611 with regression forest parameters (10,10). For the Bar's database, we also have 150 different pull intensities for the perpendicular direction. The number of sample nodes is 71,323 with (50,10) as our regression forest parameters. Figure 7 shows the comparison between BEM-based simulation result and our prediction result for two models. The result shows that our method has good estimation of fracture surfaces like the second row as well as the third row of figure. Although there is some difference between Bar's fracture surfaces at the left bottom, it seems natural to have such difference. In our naïve prediction, we only predict $\Delta\mathbf{u}$ rather than $\mathbf{u}$ which is set zero. Thus the prediction of Bar's fracture does not have bending deformation while BEM-based simulation does. This leads to such difference on fracture surfaces.

Predicting Cube's fracture under three loading modes. In previous experiments, we performed prediction based fracture simulation for all models using their respective databases. We now only use the random Cube database to test the Cube model whose loading mode for crack propagation is different from previous fracture. For the Cube database, the same database created randomly in the previous experiment described above is used.

Figure 8 represents Cube model where the Cube initiates its cracks with a planar edge-crack under loading modes I, II, and III. The three loading modes are defined in [Freund 1998] and illustrated in the first row of Figure 8. Mode I represents the crack-opening

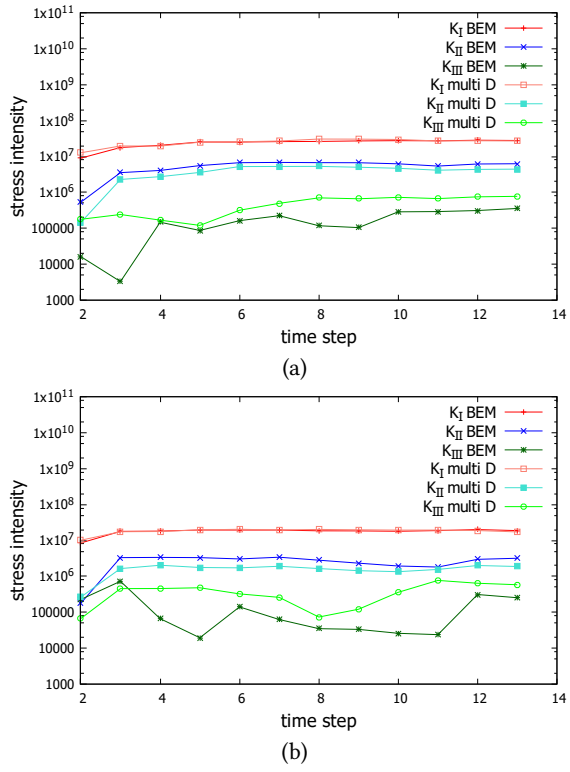

Figure 6: Comparision of stress intensities obtained by BEM-based simulation and our prediction method by using logarithmic graphs. (a) Test case for 0 degree. (b) Test case for 45 degree.

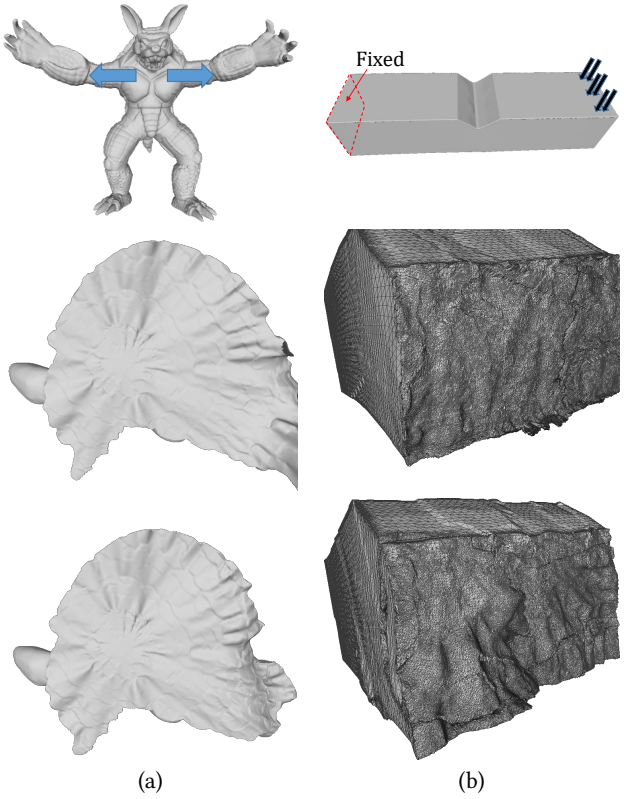

Figure 7: Comparisons between BEM-based simulation and our prediction method. (a) Results for Armadillo model. (b) Results for Bar model. From top to bottom: illustration of initial forces applied to models, fracture surfaces of two models with BEM-based simulation, fracture surfaces of two models with our prediction method.

along the perpendicular direction of the fracture surface, mode Ⅱ represents the crack-opening sliding along the normal direction of the crack-front, and mode Ⅲ represents tearing along the tangental direction of the crack-front.

The result shows that we can still use the same Cube database to well predict other situations of crack propagation which are not in database compared with BEM-based simulation. In our method, we can well predict the pattern of crack propagation in nature, which is fit to any model or situations of crack propagation.

Feature vector selection. In Section 4.1, we noted that there are three candidate feature vectors including $(K_\mathrm{I}, K_\mathrm{II}, K_\mathrm{III})$, $(K_\mathrm{I}, K_\mathrm{II}, K_\mathrm{III}, K_c)$ and $(K_\mathrm{I}, K_\mathrm{II}, K_\mathrm{III}, K_c, r)$ respectively. We explain here why we select $(K_\mathrm{I}, K_\mathrm{II}, K_\mathrm{III}, K_c)$ as our feature vector through experiments. In order to simplify the experiment process, we construct a database by only changing the magnitude of force instead of direction. The number of training samples is 9,328, and the parameters for regression forest is (10,10). Figure 9 shows the comparison between the results of BEM-based simulation and our prediction results with three features, four features, and five features. It can be seen that the fracture appearances of our prediction method with four features and five features are more similar to that of BEM-based fracture simulation than with three features. Finally, we select our feature vector with four features by a trade-off between memory and computation time.

Performance. We list the performance of BEM-based simulation and our prediction method for several models in Table 1. The results shown in this table are based on previous experiments including the prediction of the Cube's fracture with Cube database which is created randomly, and prediction of the Armadillo and Bar's fractures with their respective databases. In the "Fracture Scene" column, 0, 45 and 90 represent the direction of initial force applied to Cube.

It can be shown in the "U-com" column that the time for computing displacements can definitely be reduced compared with the BEM-based simulation. The total simulated time is not reduced through our method for the Cube. However, this result seems natural since the Cube's fracture is too simple and BEM-based simulation is quite fast. It should be noted that our method can perform better as the number of triangles in BEM mesh increases.

Fracture Scene	Tri-ini	Tri-fin	Method	Total time	U-com	Sim-frac	Other-t	frame	u/frame
Cube0	120	389	BEM	5.00s	0.09s	2.55s	2.36s	13	0.007s
	120	391	RF (random)	5.79s	0.029s	1.50s	4.26s	13	0.0022s
Cube45	120	396	BEM	5.28s	0.15s	2.69s	2.44s	13	0.011s
	120	397	RF (random)	6.74s	0.038s	1.72s	4.98s	14	0.0027s
Cube90	120	470	BEM	7.14s	0.33s	3.51s	3.5s	16	0.02s
	120	640	RF (random)	6.84s	0.040s	1.98s	4.82s	14	0.0029s
Armadillo	1000	1252	BEM	47.84s	4.89s	9.35s	33.6s	16	0.31s
	1000	1169	RF (Armadillo)	32.21s	0.21s	3.82s	28.18s	11	0.019s
Bar	416	1687	BEM	60.32s	7.49s	20.8s	32.03s	21	0.35s
	416	1825	RF (Bar)	48.02s	2.64s	11.9s	33.48s	21	0.117s

Table 1: Performance of our prediction method vs. BEM-based simulation. From left to right: (Tri-ini) The number of triangles in initial BEM mesh. (Tri-fin) The number of triangles in final BEM mesh. (Method) Method of fracture including BEM-based simulation and regression forest-based prediction method (RF). (Total time) The total simulated time during the whole fracture process. (U-com) The time for computing CODs. (Sim-frac) The total time for simulating fractures. (Other-t) The total time for other processes like reading a model, writing to disk, loading regressors with our method etc. (frame) The number of frames for simulation. (u/frame) The time for computing displacement per frame.

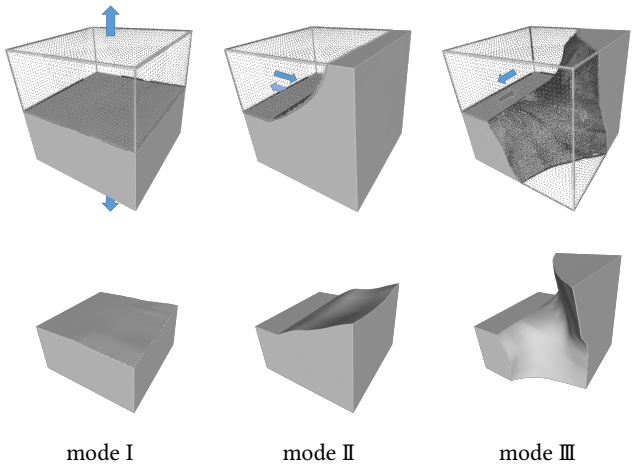

Figure 8: Comparison results between BEM method and our prediction method under three loading modes. The first row represents BEM-based simulation results with initial loading modes, and the second row represents their cossepond-ing predction results.

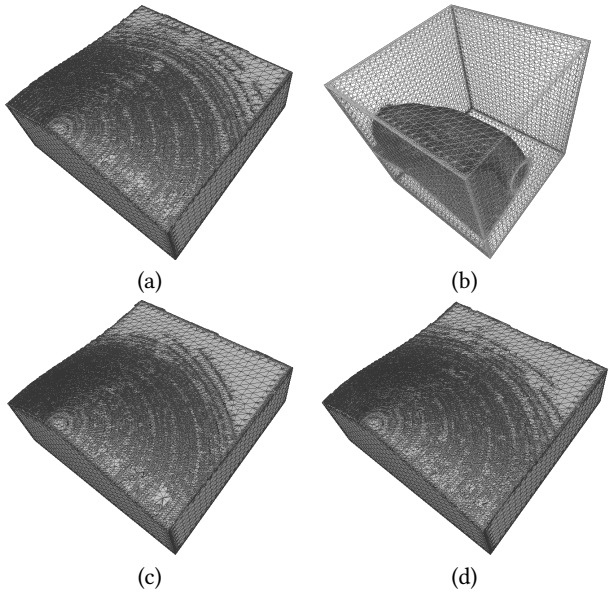

Figure 9: (a) BEM-based simulation result for Cube. (b)-(d) Prediction results with three features, four features, and five features respectively. All experiments are performed with the same initial force.

6 CONCLUSION AND FUTURE WORK

In this paper, we propose a machine learning scheme that achieves fast approximation of crack-opening displacement (COD) instead of solving linear equations in the propagation of cracks. We designed a set of features including stress intensities and toughness at the current frame which have a strong influence on CODs at the next frame in our naïve prediction, and trained a regressor capable of predicting the projections of CODs of the next frame. Our method reduces the time for computing CODs compared with BEM-based simulation. The time cost can be reduced considerably especially when the resolution of initial BEM mesh increases. Our approach shows a great potential for replacing the traditional BEM solver, especially when the time cost is more important than the accuracy of fracture simulations, like in computer games. This is the first time a machine learning scheme is used in BEM-based fracture simulation, which should serve as useful reference for other researchers.

In our future work, we will try to combine several databases from different models and use it to test several different models. In addition, there are improvements that can be made such as acceleration of the mesh update process. Another interesting direction is to try to use different machine learning approaches such as Convolutional Neural Network (CNN) for our regression. Finally, we

will attempt to combine fracture animations with the rigid body dynamics engine.

REFERENCES

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

Leo Breiman. 2001. Random Forests. *Machine Learning* 45, 1 (01 Oct 2001), 5–32. https://doi.org/10.1023/A:1010933404324

A Frangi, G Novati, R Springhetti, and M Rovizzi. 2002. 3D Fracture Analysis by the Symmetric Galerkin BEM. *Computational Mechanics* 28, 3 (2002), 220–232. https://doi.org/10.1007/s00466-001-0283-x

Lambert B. Freund. 1998. *Dynamic Fracture Mechanics.* Cambridge University Press.

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 (Feb. 2013), 201–209. https://doi.org/10.1109/TVCG.2012.121

Institute of Applied Mechanics Graz University of Technology. 2016. HyENA - Hyperbolic and Elliptic Numerical Analysis. https://www.tugraz.at/en/institutes/am-bm/research/software/. (2016).

David Hahn and Chris Wojtan. 2015a. FractureBEM. https://github.com/david-hahn/FractureBEM. (2015). Accessed: August 11, 2017.

David Hahn and Chris Wojtan. 2015b. 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. *Vis. Comput.* 14, 3 (01 July 1998), 126–137. https://doi.org/10.1007/s003710050128

Anthony R. Ingraffea and Paul A. Wawrzynek. 2003. Finite Element Methods for Linear Elastic Fracture Mechanics. *Comprehensive Structural Integrity* 3 (2003), 1–88. https://doi.org/10.1016/B0-08-043749-4/03007-X

Lars Kielhorn. 2009. *A Time-Domain Symmetric Galerkin BEM for Viscoelastodynamics.* Verlag der Techn. Univ. Graz.

L'ubor Ladický, SoHyeon Jeong, Barbara Solenthaler, Marc Pollefeys, and Markus Gross. 2015. Data-driven Fluid Simulations Using Regression Forests. *ACM Trans. Graph.* 34, 6, Article 199 (Oct. 2015), 9 pages. https://doi.org/10.1145/2816795.2818129

Oleg Mazarak, Claude Martins, and John Amanatides. 1999. Animating Exploding Objects. In *Proceedings of the 1999 Conference on Graphics Interface '99.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 211–218.

David Mould. 2005. Image-guided Fracture. In *Proceedings of Graphics Interface 2005 (GI '05).* Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 219–226. http://dl.acm.org/citation.cfm?id=1089508.1089545

Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. 2013. Real Time Dynamic Fracture with Volumetric Approximate Convex Decompositions. *ACM Trans. Graph.* 32, 4, Article 115 (July 2013), 10 pages. https://doi.org/10.1145/2461912.2461934

Matthias Müller, Leonard McMillan, Julie Dorsey, and Robert Jagnow. 2001. Real-time Simulation of Deformation and Fracture of Stiff Materials. In *Proceedings of the Eurographic Workshop on Computer Animation and Simulation.* Springer-Verlag New York, Inc., New York, NY, USA, 113–124. http://dl.acm.org/citation.cfm?id=776350.776361

Michael Neff and Eugene Fiume. 1999. A Visual Model for Blast Waves and Fracture. In *Proceedings of the 1999 Conference on Graphics Interface '99.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 193–202. http://dl.acm.org/citation.cfm?id=351631.351686

Alan Norton, Greg Turk, Bob Bacon, John Gerth, and Paula Sweeney. 1991. Animation of Fracture by Physical Modeling. *Vis. Comput.* 7, 4 (July 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., New York, NY, USA, 137–146. https://doi.org/10.1145/311535.311550

Machine Perception and Chubu University Robotics Group. 2014. Random Forest and Regression Forest. http://mprg.jp/tutorials/random-regression. (2014). Accessed: August 11, 2017.

Saty Raghavachary. 2002. Fracture Generation on Polygonal Meshes Using Voronoi Polygons. In *ACM SIGGRAPH 2002 Conference Abstracts and Applications (SIGGRAPH '02).* ACM, New York, NY, USA, 187–187. https://doi.org/10.1145/1242073.1242200

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

Jonathan Tompson, Kristofer Schlachter, Pablo Sprechmann, and Ken Perlin. 2016. Accelerating Eulerian Fluid Simulation With Convolutional Networks. (2016). arXiv:arXiv:1607.03597

Cheng Yang, Xubo Yang, and Xiangyun Xiao. 2016. Data-driven Projection Method in Fluid Simulation. *Computer Animation and Virtual Worlds* 27, 3-4 (2016), 415–424. https://doi.org/10.1002/cav.1695 cav.1695.

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