Predicting Brittle Fracture Surface Shape From Versatile Database

Yuhang Huang, Yonghang Yu, Takashi Kanai

The University of Tokyo

Graduate School of Arts and Sciences

3-8-1, Komaba, Meguro-ku, Tokyo, 153-8902, Japan

email: 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 naive 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. The databases generated by our method are versatile, and can be applied to general situations and different models.

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

1 Introduction

Rigid body fracture simulation is playing an increasingly 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.

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 [1]. 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 [2], 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 [3] 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.

Compared with our previous work [4], this paper attempts to extend our method to create a versatile database, which can be applied to more general situations and models different in nature. Generating the database of complicated models is expensive and time-consuming due to the need to conduct simulations and training of regression forests. Our aim is to gener-

ate a database from simple models like Cube models, which can be applied to the simulation of the fractured surfaces of complex models like that of an Armadillo. To achieve this aim, we redesign the feature vector to realize scale independence in models with different sizes, and regenerate databases with the new feature vector.

2 Related work

2.1 Geometry-based Fracture Animation

Terzopoulos and Fleischer [5] 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 [6] describes a method of generating fracture fragments under the principle of Voronoi tessellation. Neff and Fiume [7] 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 [8] 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. [9] 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 in pairs. Norton et al. [10] modeled objects from simple lattice cubic cells, then generated fractures by breaking the link between cubes. Hirota et al. [11] 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. [12] 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 [13] proposed brittle fracture simulations with the finite element method (FEM). Later,

O'Brien et al. [14] again presented ductile fracture as well. Although FEM can well define the orientation 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. [15] proposed a hybrid method where static analysis is independent of time-step allowing simulating of fractures of objects in real-time. Bao et al. [16] treated material as fully rigid body to solve issues introduced by small time step restrictions. Glondu et al. [17] 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 [1] 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 [2], but a linear system still has to be solved. Zhu et al. [18] 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 [19]. The projection step is a time-consuming step in grid-based fluid simulation. Yang et al. [20] 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 [21].

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 Wojitan [1]. 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 [22]. 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 [22]).

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 [23], 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 [24],

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

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 $\Delta\mathbf{u}_{\text{I}}$, $\Delta\mathbf{u}_{\text{II}}$, and $\Delta\mathbf{u}_{\text{III}}$.

Effective stress intensities are then calculated as ${K_{\text{eff}}}^2 = {K_{\text{I}}}^2 + {K_{\text{II}}}^2 + {K_{\text{III}}}^2/(1-\nu)$, 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 [25] for details). Cracks propagate if $K_{\text{eff}} \geq K_c$. The detailed crack propagation including propagation speed and direction is described in [1].

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 [3]. 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 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 naive prediction is performed at the same coordinate system.

Furthermore, we apply normalization rules to K_{I}, K_{II}, K_{III} and $\Delta\mathbf{u}_{\mathrm{I}}$, $\Delta\mathbf{u}_{\mathrm{II}}$, $\Delta\mathbf{u}_{\mathrm{III}}$ in the local coordinate system to reduce the influence of the BEM mesh size. The results of several experiments indicate that the distance unit $\Delta\mathbf{u}$ relies on the size of models. Error will occur when we apply a small-size database to big-size models. Based on Equation (3), there is

unit dimension difference of $\frac{1}{\sqrt{r}}$ between K_{I} and $\Delta\mathbf{u}_{\mathrm{I}}$ (also K_{II} and $\Delta\mathbf{u}_{\mathrm{II}}$, K_{III} and $\Delta\mathbf{u}_{\mathrm{III}}$) after excluding dimensionless quantities. Therefore, we apply division by $\sqrt{\overline{r}}$ to K_{I}, K_{II}, K_{III}, and by $\overline{r}$ to $\Delta\mathbf{u}_{\mathrm{I}}$, $\Delta\mathbf{u}_{\mathrm{II}}$, $\Delta\mathbf{u}_{\mathrm{III}}$, where $\overline{r}$ refers to the resolution of the BEM mesh. Distance r for which the crack-front has propagated will change in each time step, so that the distance of BEM mesh resolution $\overline{r}$ is appropriate for normalization. The material toughness K_c does not differ amongst models with different resolutions, so we count K_c out when applying normalization rules.

With predicted projections of CODs which are multiplied by $\overline{r}$, 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 [1].

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 [3] 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,

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

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 $n-1$, $\overline{r}$ refers to the resolution of the BEM mesh among all frames, and $\Delta \mathbf{u}_i^{(n)}$ 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 five different databases, namely Cube database, Cylinder 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 [26] provided by the authors of [1] and HyeNA library [27] provided by the Institute of Applied Mechanics, Graz University of Technology. In the training and prediction stage, we also use the source code [28] 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).

All experiments are simulated based on the following parameters: Young's modulus $E = 3.1 \times 10^9 Pa$, Poissons ratio $\nu = 0.327$, density $\rho = 1200 kg/m$, tensile strength (principal stress at surfaces) $S_c = 7.6 \times 10^7 Pa$ and fracture toughness (stress intensity at crack-tips) $K_c = 1 \times 10^6 Pa$. The size of the Armadillo model and Bar model is approximately 100 times bigger than the size of the Cube and Cylinder models. The edge length of the Cube model is $1m$ and the diameter of the Cylinder model is $1m$.

Prediction of Cube fractures. As discussed in Section 4.3, the training samples are created by changing the direction and magnitude of forces applied to objects. For directions and magnitudes of forces applied to the 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 the force at one side. At another side, there is a force which has the same pull intensity but

acts from a different direction. Both 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 the Cube database, we have 992,976 training samples. The force is applied on the same face of a surface every time.

We first test the 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 (a) and (b) of the figure, one half of the split Cube is shown. In the third column (c), 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.

In the third column (c), the slopes of the fracture surfaces differ between the BEM-based simulation results and our data-driven prediction results. Since the experiments in this research focused on the pulling scene, our parameters of force direction were mainly set to 0 degrees and the surrounding direction. This always realizes stable fracture surface shapes. Furthermore, in the BEM-based simulation results where the force direction was set to 90 degrees, the shape of fracture surfaces changed suddenly, even though there was

only a small difference in the force parameters (initial boundary conditions) between 89 and 90 degrees. Therefore, the prediction of the database based on the results with the force direction set to 90 degrees does appear to be similar to the BEM-based result as the predictions with 0 and 45 degree forces are.

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. Furthermore, according to Table 1, which shows the statistics for the coefficients of stress intensities obtained by the BEM-based fracture simulation before the training stage, the coefficient of variation of K_{III} of training samples is larger than that of K_{I} and K_{II}. Since the Cube database is trained by these training samples, it seems natural that the prediction of the Cube database K_{III} has larger variation than the prediction of the Cube databases K_{I} and K_{II}.

Prediction of Cylinder fractures. To evaluate the reproducibility of training samples obtained by random forces generated for different shapes, we created a database for a Cylinder

model by changing both the force magnitudes and directions, which is the same method as creating the Cube database.

For the Cylinder database, forces are simultaneously applied to the top and the bottom of a round surface area of a Cylinder model. The number of sample nodes of the Cylinder database is 492,373 with regression forest parameters (50,10). For the Cylinder database, we apply forces with the same magnitude as the Cube database but from different directions as shown in Figure 4. Since the Cylinder model does not crack in both BEM-based simulation and prediction when applied with forces from directions whose $\alpha > 80$ degree and $\beta = 0$ degree, we simply test the Cylinder database using the Cylinder model with force directions whose $\alpha = 0, 45, 80$ degree and $\beta = 0$ degree. Figure 7 shows the comparison between BEM-based simulation result and our prediction result for the Cylinder model. Through the Figures 7 (a) to (c), the prediction results using our method in the bottom row reproduce the slope of the fractured surface and the height from the bottom faces to the fractured surfaces. Also, the details of the fractured surface patterns are similar.

This result shows that the prediction results for the Cylinder modes reproduce the typical characteristics of a fractured surface. Thus our method can be applied to both Cube and Cylinder models, and fractured surfaces can be well approximated, as shown in the middle row as well as the bottom row of the figures.

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 models by changing only the force magnitudes for the same direction. For the Armadillo database, we simulated with 150 different pull intensities for the horizontal direction. The number of sample nodes is 12,611 with regression forest parameters (10,10). On the other hand, for the Bar database we simulated with 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 8 shows the comparison between BEM-based simulation result and our prediction result for the Bar model. Figures 10 (a) to (c) show the comparison between BEM-based simulation result and our prediction result for the Armadillo model. The results confirm that our method is able to accurately estimate fractured surfaces as shown by (b) and (c) of Figures 8 and 10 respectively. Although there is some difference between Bars fractured surfaces at the left bottom, it seems natural to have such differences. In our naive prediction, we only predict $\Delta\mathbf{u}$ rather than $\mathbf{u}$ which is set zero. Thus, the prediction of Bars fractures does not have bending deformation after the second time-step frame while the BEM-based simulation does, resulting in these differences in the fractured surfaces. Nevertheless, we still carry out the BEM-based simulation at the first time-step frame in our method. Initial deformation still occurs on the other part of the Bar model, such as the top slope which is not touched by the crack surface. To interpolate the crack surface and the other parts of the model during visualization, the prediction results of such deformations at the top slope with our prediction method differ slightly from BEM-based simulation result.

Predicting Cube fractures 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 9 shows a 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 [25] and illustrated in the first row of Figure 9. Mode I represents the crack-opening 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 use the same Cube database to predict three loading modes. Even though the prediction results of mode II and mode III shown in the second and third columns of Figure 9 are not the same as the BEM-based simulation results, the prediction results show that the three loading modes tend to demonstrate the expected crack propagation behavior, as shown in the first row of Figure 9, indicating good prediction performance. It should be emphasized that the Cube database prediction method cannot predict exactly the same shapes as the BEM-based simulation method without a specific database. In order to completely predict BEM-based simulation results with three loading modes, a specific database trained for the loading situation of each of the three modes is needed. It should also be noted that the Cube database prediction method using only training samples with random

pick-up direction forces can also represent the expected crack propagation behavior, which is the principal behavior in fracture mechanism. Generally, while the force is applied, a displacement will be created with the same direction as the force's. Since the feature vector has been designed to reduce the influence of both direction and length as described in Section 4.2. The Cube database has been trained to predict the correlation of stress intensities and displacements. Therefore, there is potential to use the same Cube database to well predict other situations.

Predicting Armadillo fractures with other databases. In the previous experiment, we found that the same Cube database may be used for well predicting other situations. Therefore, we use the Cube and Cylinder databases to predict Armadillo models, and compare the results of the Cube and Cylinder databases with those of the Armadillo database. The results shown in Figure 10 are computed for the Armadillo model with the same initial force and respective databases.

For the Cube database, the same database created randomly in the previous experiment described above is used. For the Cylinder database, we use the databases generated in the experiments above. Figure 10 (a) shows the initial force that we used to test the databases. Figures 10 (b) to (e) show the results of BEM-based simulations, prediction using the Armadillo database, prediction using the Cube database, and prediction using the Cylinder database.

The results in Figure 10 (e) curve outwards in the whole surface region, which shows

that the shape of the Armadillos fractured surface generated by the Cylinder database cannot be well predicted as the Cube database. Even though the fractured surfaces predicted by the Armadillo database in Figure 10 (c) is more similar to the BEM-based fractured surface in Figure 10 (b) than that predicted by other databases in terms of the patterns of fracture surface. All of the results in 10 (c) to (e) reproduce acceptable patterns and shapes of the Armadillo fractured surface in 10 (b). The results show that we can still use the same Cube database to well predict other crack propagation situations which are not in the database compared with BEM-based simulation. With our method, we can well predict the nature of the pattern of crack propagation, which fits any models or situations of crack propagation.

Even though the Cube database works well for predicting the fractured surfaces of Armadillo models, further analysis of the Cube database is required to clarify its mechanism of versatility, and we will conduct this in future work.

Comparisons of crack propagation process between Armadillo and Cube fractures. In previous experiments, we compared the final surfaces between BEM-based simulation results and our prediction results with several models like Cube and Armadillo models. However, the process of crack propagation must also be compared and discussed. Therefore, we used the prediction results obtained from previous experiments using Cube and Armadillo databases for comparing BEM-based simulations in each timestep frame, and selected three frames for Cube and Armadillo models respectively.

Figure 11 shows the comparison between the two crack-propagating process results

shown in the second and the third rows in Figure 5 (a). Figure 12 shows the comparison between the two crack-propagating process results of (b) and (c) in Figure 10.

According to the results shown in Figure 11 and Figure 12, we found that the speed of crack propagation and the shape of fracture surfaces are similar between prediction results and BEM-based results in each timestep frame for the Cube and Armadillo models respectively.

The number of frames until fracture is completed in both prediction results and BEM-based results for Cube experiments shown in Figure 5 (a) are 13. We picked the 1^{st}, 6^{th}, and 11^{th} frames to represent the process of crack propagation of Cube database experiments in Figure 11.

In the Armadillo experiments shown in Figure 10 (b) and (c), the number of frames until fracture is completed in BEM-based results is 16, and the number of frames in prediction results is 11. However, the speed of crack propagation of these two experiments is similar through the preceding 11 frames between BEM-based results and prediction results. The results show that our proposed method can reproduce the speed of crack propagation in prediction. Since the crack propagation of the Armadillo model after 5^{th} frame is extremely slow and any difference between BEM-based and prediction fractures after 10^{th} frame is difficult to find without detailed observation, the reason of earning different numbers of frames in BEM-based simulation and our prediction method is not the speed of crack propagation, but the stopping time of crack propagation. Even though the determination mechanism of stopping of BEM-based simulation is the same as that of our prediction method, feedback

cannot be controlled if the BEM linear system is not used in our prediction method, which causes the prediction process to stop crack propagation faster than the BEM-based simulation method.

In conclusion, our prediction method can reproduce the speed of crack propagation. The number of frames until fracture occurs with our prediction method may not be the same as that in BEM-based simulation, even though the main process of crack propagation occurs in the preceding several frames for both our prediction results and BEM-based simulation results.

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 13 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.

Training sample selection when generating databases. We explain here what should be

noted when creating a database.

The results of past experiments have shown that the prediction of cracks is not always successful with the application of an initial force in BEM-based fracture simulations. Predictions often fail in the use of databases of training samples containing situations of the generation of a large number of small cracks or situations where cracking is non-stop. Most of the BEM-based simulations applying force directions (e.g. the direction whose $\alpha = 90$ degree and $\beta = [0, 360]$ degree) do not cause fractured surfaces or cause small cracks in Cube and Cylinder models. In many cases, these simulations can simulate cracks over 80 frames or non-stop cracking situations, while most simulations normally cause cracks of only 10 frames in our test configuration. Such cases which contain small cracks and do not stop cracking are outside the scope of our interests in the prediction of patterns and shapes of fractured surfaces. The resultant training samples are chaotic, and will influence the other traning samples in our database training set. These cases which are not acceptable for predicting shapes and patterns of fractured surfaces should be excluded from training sample sets.

For the Cube and Cylinder models, BEM-based simulation with force directions whose $\alpha = 90$ degree and $\beta = [0, 360]$ cracks as shown in Figure 5. We stop cracking when there are more than 31 frames, while most of the simulations stop in 20 frames. For the Cube model, there are some BEM-based unnatural simulation cracks in small cracks with force directions whose α is nearly to 90 degree and $\beta = [0, 360]$. We exclude all these training samples with such force directions. It is important to set the appropriate limitation of force

directions and maximum simulation frames should be noted when generating database.

Though the versatile database created by our method predicts well in most situations and models, there are many optional cracking configurations like the maximum number of initial crack points and the limitation option of face cracking or line cracking. We should still train a database adapting the specific model and cracking configuration every time a highly similar fractured surface with better details compared with the BEM-based simulations is needed.

Parameter settings when training databases. We explain here what should be noted when setting parameters before training a database.

Parameters of regression forest [3] usually depend on the number of training samples. Trees with larger leaves can explain situations predicted by regression forests better. Taking t to be the number of trees and d the maximal depth of tree, a set of two parameters (t, d) can determine the maximum total number of leaves that the regression forests can keep. We try to set an appropriate (t, d), which makes the maximal total number of leaves larger than the number of training samples. After executing several experiments by changing the parameter set (t, d) with different numbers of training samples of Cube model from $10,000$ to $1,000,000$ and by evaluating the reproducibility with the same Cube model, the two main tendencies we found in parameter setting are; (1) the larger the number of t, the more stably will the database be trained, and (2) an appropriate number of d should be set for preventing overfitting depending on the number of training samples like $d = 10$ or $d = 15$.

In conclusion, we set several (t, d) and evaluated the reproducibility of the databases. Finally, we chose "RF(Cube) - 1" as the random forest of Cube model, and "RF(Cylinder) - 1" as the random forest of Cylinder model in Table 2.

Performance. We list the performance of BEM-based simulation and our prediction method for several models in Table 3. 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, the prediction of the Armadillo and Bar's fractures with their respective databases, and the prediction of the Armadillo fracture with the Cube and Cylinder databases which are created randomly. 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 simulation time does not decrease through our method for Cube. However, this result seems natural since the Cube's fracture is too simple and BEM-based simulation is quite fast.

In rows "Cube45", "Cube90" and "Armadillo", we found that the number of frames in our proposed method is different compared with that in BEM-based simulation. This is because the number of frames changes dramatically when we just adjust force parameters slightly. Since different trained database causes different number of frames, it seems hard to compare scenes with different number of frames. Therefore we design an index of "u/frame", which means the time for computing displacements per frame, to compare

the time in different scenes. It can be seen in the "u/frame" column that the time for computing displacements per frame can definitely be reduced compared with the BEM-based simulation.

In Table 3, results also show that the simulation time of the Bar model is considerably longer than that of the Armadillo model. This is because the force loaded on Bar model is in shear direction, therefore the speed of cracking differs according to the crack directions. Different cracking speed causes different number of triangles generated on fracture surfaces. According to the columns of "Tri-fin" and "Tri-ini", the Bar model has more triangles than the other models. The simulation time of the Bar model is thus longer than the other models. Since the time of COD prediction in our method increases linearly when the triangles of fracture mesh increase, the prediction time of the Bar model is 10 times longer than that of the Armadillo model when the number of triangles of the Bar model is 10 times more than that of the Armadillo model.

It should be noted that our method can perform better as the number of triangles in BEM mesh increases.

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 naive 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.

Since the results on computational performance show that the reduction of simulation time by our method is quite limited, our method still cannot be put to practical application for computer games, etc. However, since the time for computing CODs has been dramatically reduced and the acquisition of displacements accelerated 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. For practical applications, there is a need to not only reduce the time for computing CODs, but also the time of the mesh generation process, which is included in the simulation of fractures. Also, our method is the first method applying machine learning approach to BEM-based fracture simulation, which should serve as useful reference for other researches.

In our future work, we will thoroughly analyze the different regression forest trees of Cube and Cylinder databases to determine the Cube database can perform better than the other databases when predicting the fractured surfaces of Armadillo models. Furthermore, we will also try to extend our method to recompute the BEM linear system at the midtime,

to solve repeated cracking and improve prediction accuracy. 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

[1] David Hahn and Chris Wojtan. High-resolution brittle fracture simulation with boundary elements. *ACM Trans. Graph.*, 34(4):151:1–151:12, July 2015.

[2] David Hahn and Chris Wojtan. Fast approximations for boundary element based brittle fracture simulation. *ACM Trans. Graph.*, 35(4):104:1–104:11, July 2016.

[3] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[4] Yonghang Yu, Yuhang Huang, and Takashi Kanai. Data-driven approach for simulating brittle fracture surfaces. In *ACM SIGGRAPH ASIA 2017 Workshop: Data-Driven Animation Techniques (D2AT)*, pages 3:1–3:8, 2017.

[5] Demetri Terzopoulos and Kurt Fleischer. Modeling inelastic deformation: Viscolelasticity, plasticity, fracture. *SIGGRAPH Comput. Graph.*, 22(4):269–278, June 1988.

[6] Saty Raghavachary. Fracture generation on polygonal meshes using voronoi polygons. In *ACM SIGGRAPH 2002 Conference Abstracts and Applications*, SIGGRAPH '02, pages 187–187, New York, NY, USA, 2002. ACM.

[7] Michael Neff and Eugene Fiume. A visual model for blast waves and fracture. In *Proceedings of the 1999 Conference on Graphics Interface '99*, pages 193–202, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[8] David Mould. Image-guided fracture. In *Proceedings of Graphics Interface 2005*, GI '05, pages 219–226, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2005. Canadian Human-Computer Communications Society.

[9] Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. Real time dynamic fracture with volumetric approximate convex decompositions. *ACM Trans. Graph.*, 32(4):115:1–115:10, July 2013.

[10] Alan Norton, Greg Turk, Bob Bacon, John Gerth, and Paula Sweeney. Animation of fracture by physical modeling. *Vis. Comput.*, 7(4):210–219, July 1991.

[11] Koichi Hirota, Yasuyuki Tanoue, and Toyohisa Kaneko. Generation of crack patterns with a physical model. *Vis. Comput.*, 14(3):126–137, July 1998.

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

[13] James F. O'Brien and Jessica K. Hodgins. Graphical modeling and animation of brittle fracture. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 137–146, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[14] James F. O'Brien, Adam W. Bargteil, and Jessica K. Hodgins. Graphical modeling and animation of ductile fracture. *ACM Trans. Graph.*, 21(3):291–294, July 2002.

[15] Matthias Müller, Leonard McMillan, Julie Dorsey, and Robert Jagnow. Real-time simulation of deformation and fracture of stiff materials. In *Proceedings of the Eurographic Workshop on Computer Animation and Simulation*, pages 113–124, New York, NY, USA, 2001. Springer-Verlag New York, Inc.

[16] Zhaosheng Bao, Jeong-Mo Hong, Joseph Teran, and Ronald Fedkiw. Fracturing rigid materials. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):370–378, March 2007.

[17] Loeiz Glondu, Maud Marchal, and Georges Dumont. Real-time simulation of brittle fracture using modal analysis. *IEEE Transactions on Visualization and Computer Graphics*, 19(2):201–209, February 2013.

[18] Yufeng Zhu, Robert Bridson, and Chen Greif. Simulating rigid body fracture with surface meshes. *ACM Trans. Graph.*, 34(4):150:1–150:11, July 2015.

[19] L'ubor Ladický, SoHyeon Jeong, Barbara Solenthaler, Marc Pollefeys, and Markus Gross. Data-driven fluid simulations using regression forests. *ACM Trans. Graph.*, 34(6):199:1–199:9, October 2015.

[20] Cheng Yang, Xubo Yang, and Xiangyun Xiao. Data-driven projection method in fluid simulation. *Computer Animation and Virtual Worlds*, 27(3-4):415–424, 2016. cav.1695.

[21] Jonathan Tompson, Kristofer Schlachter, Pablo Sprechmann, and Ken Perlin. Accelerating eulerian fluid simulation with convolutional networks, 2016.

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

[23] A Frangi, G Novati, R Springhetti, and M Rovizzi. 3d fracture analysis by the symmetric galerkin BEM. *Computational Mechanics*, 28(3):220–232, 2002.

[24] Anthony R. Ingraffea and Paul A. Wawrzynek. Finite element methods for linear elastic fracture mechanics. *Comprehensive Structural Integrity*, 3:1–88, 2003.

[25] Lambert B. Freund. *Dynamic Fracture Mechanics*. Cambridge University Press, 1998.

[26] David Hahn and Chris Wojtan. Fracturebem. `https://github.com/david-hahn/FractureBEM`, 2015. Accessed: August 11, 2017.

[27] Institute of Applied Mechanics Graz University of Technology. Hyena - hyperbolic and elliptic numerical analysis. https://www.tugraz.at/en/institutes/am-bm/research/software/, 2016.

[28] Machine Perception and Chubu University Robotics Group. Random forest and regression forest. http://mprg.jp/tutorials/random-regression, 2014. Accessed: August 11, 2017.

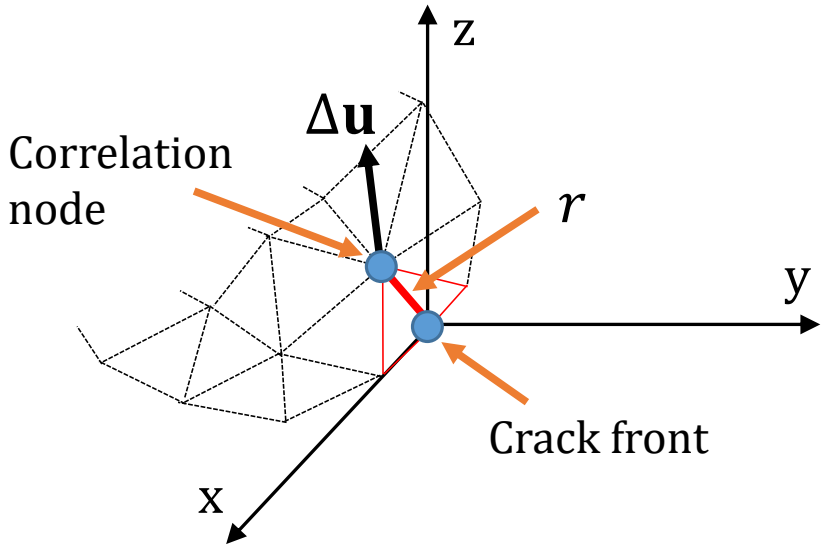


Figure 1: Displacement correlation technique. $\Delta\mathbf{u}$ is evaluated on correlation node of crack-front.

Statistics coefficient	K_{I} Training Samples	K_{II} Training Samples	K_{III} Training Samples
Median	$14,521,100$	$-1,400,000$	$81,253.2$
Average	$16,614,840.62$	$-2,373,040.041$	$65,368.94$
Dispersion	$1.9919E^{+14}$	$2.6533E^{+13}$	$3.2648E^{+12}$
Standard deviation	$14,113,317.33$	$5,150,967.276$	$1,806,866.553$
Coefficient of variation	0.8494	-2.1706	27.6411

Table 1: **Statistics about coefficients of stress intensities $K_{\mathbf{I}}, K_{\mathbf{II}}, K_{\mathbf{III}}$ obtained by BEM-based simulation before training stage.** (Training Samples) The raw training samples data collected by BEM-based simulation with Cube model. (Coefficient of variation) Standard deviation divided by average.

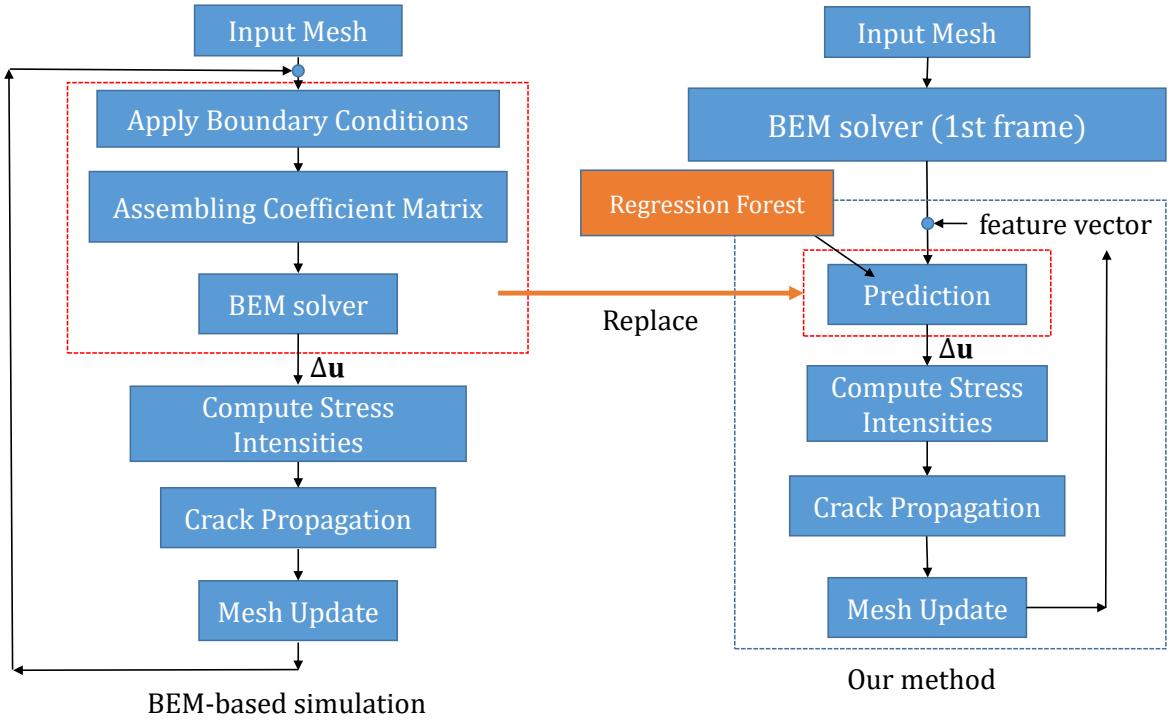


Figure 2: Left: Flowchart of BEM-based fracture simulation system. Right: Flowchart of our prediction system.

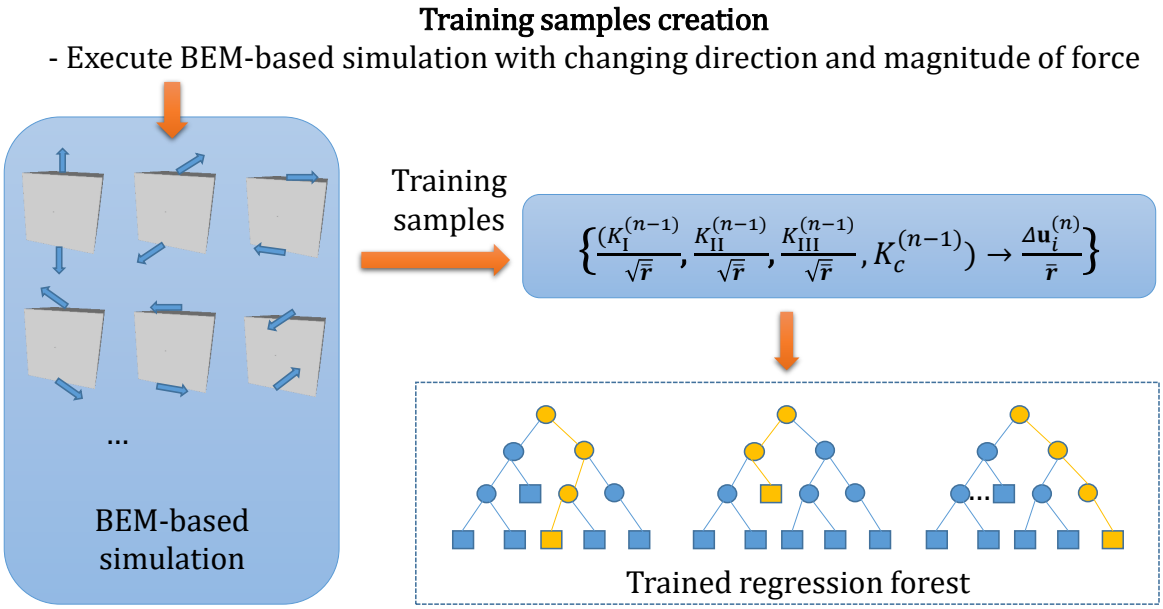


Figure 3: Creation of training samples and the construction of regression forest.

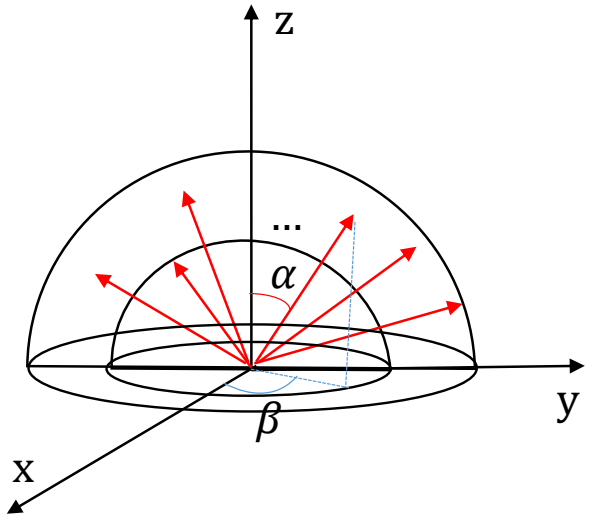
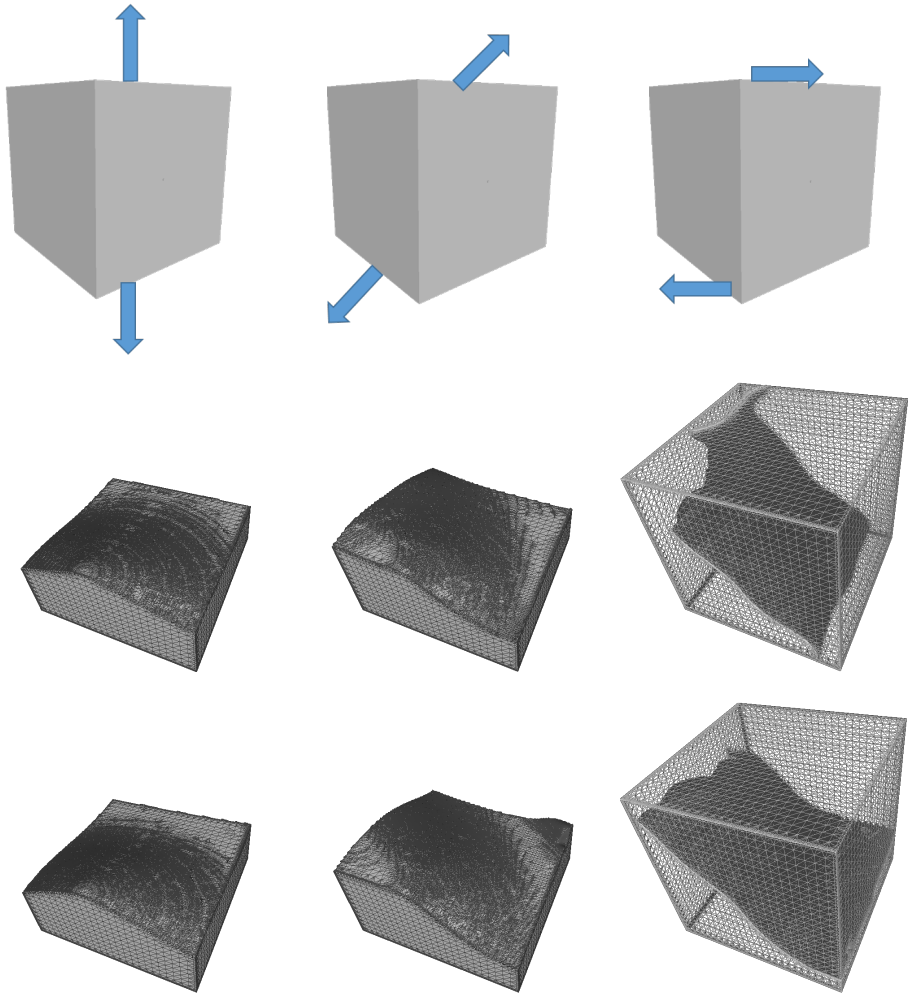
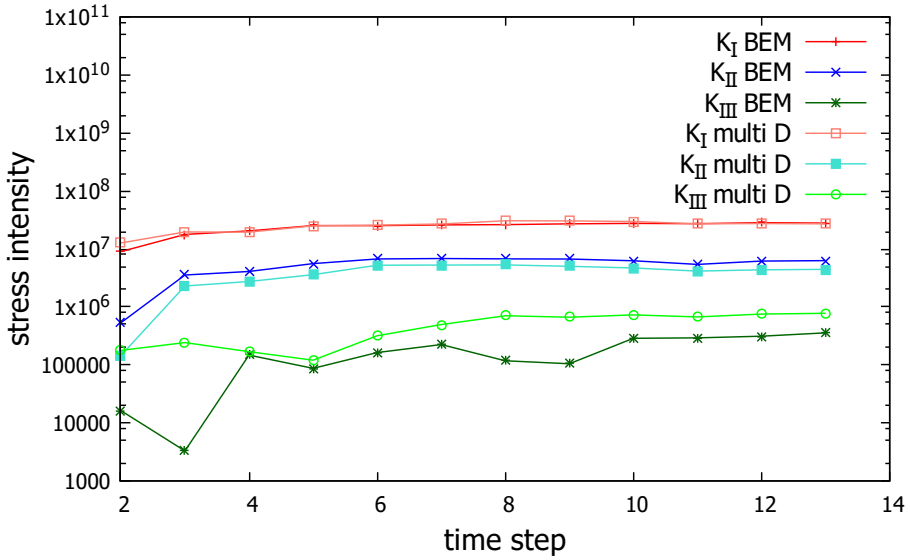


Figure 4: Random force creation. α is the slope angle and β is the rotation angle.

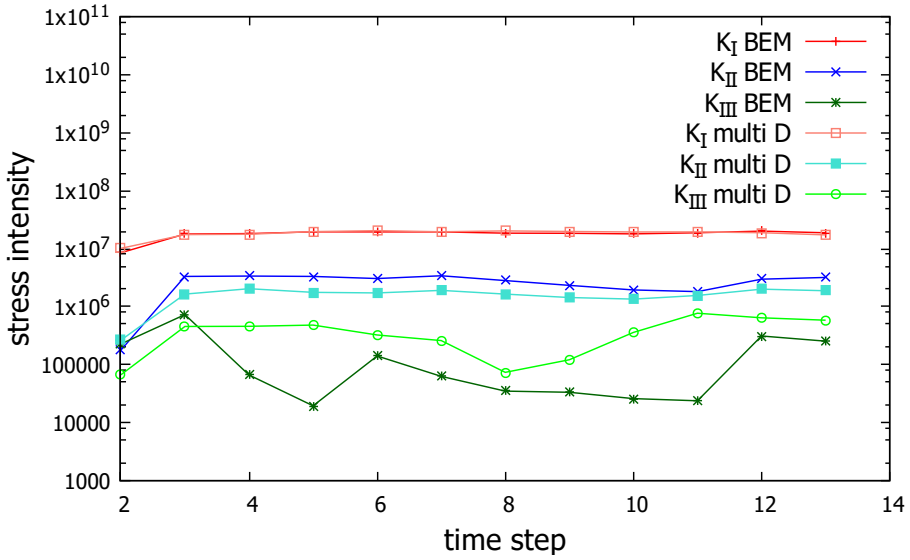


(a) (b) (c)

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.

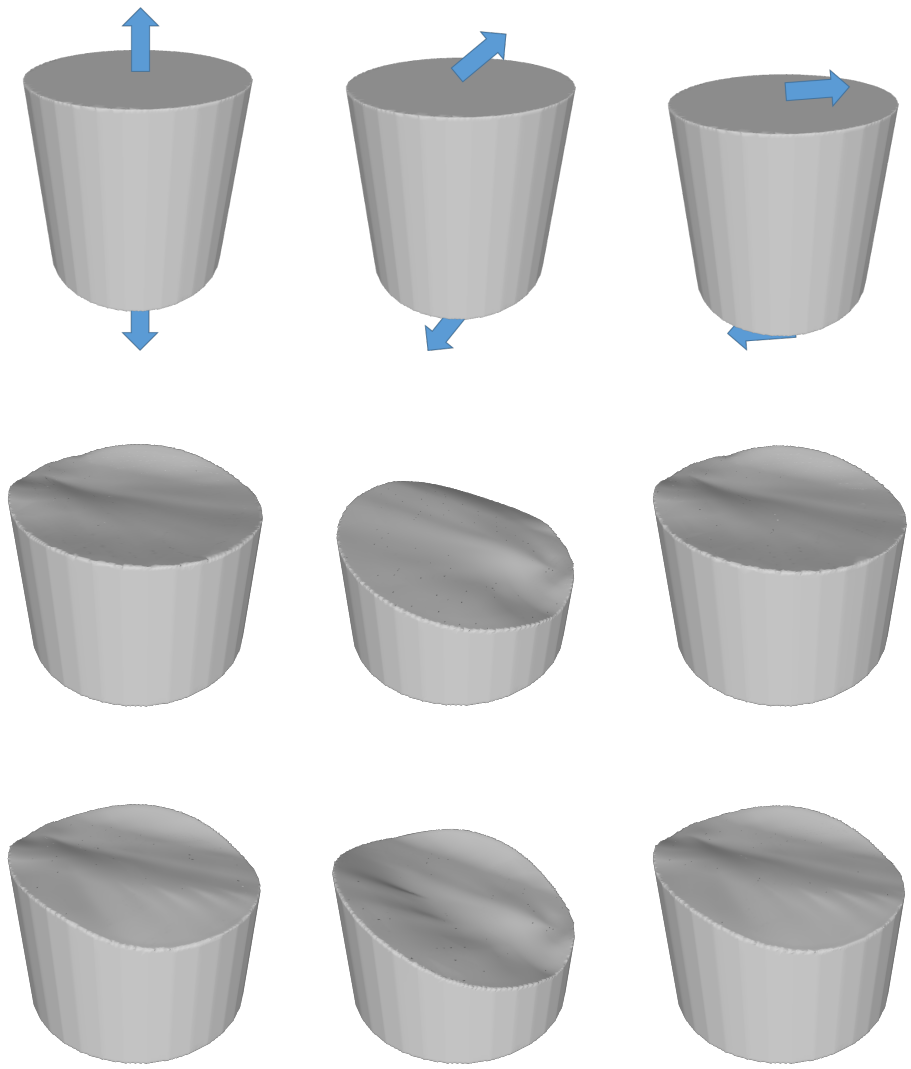


(a)



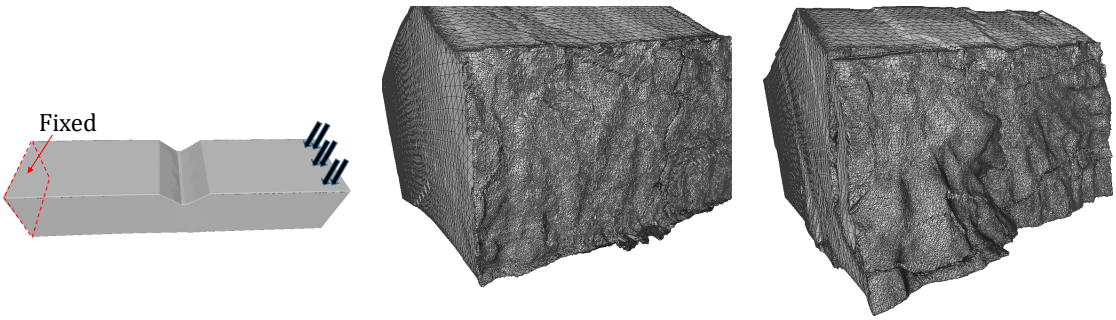
(b)

Figure 6: Comparison 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 degrees.



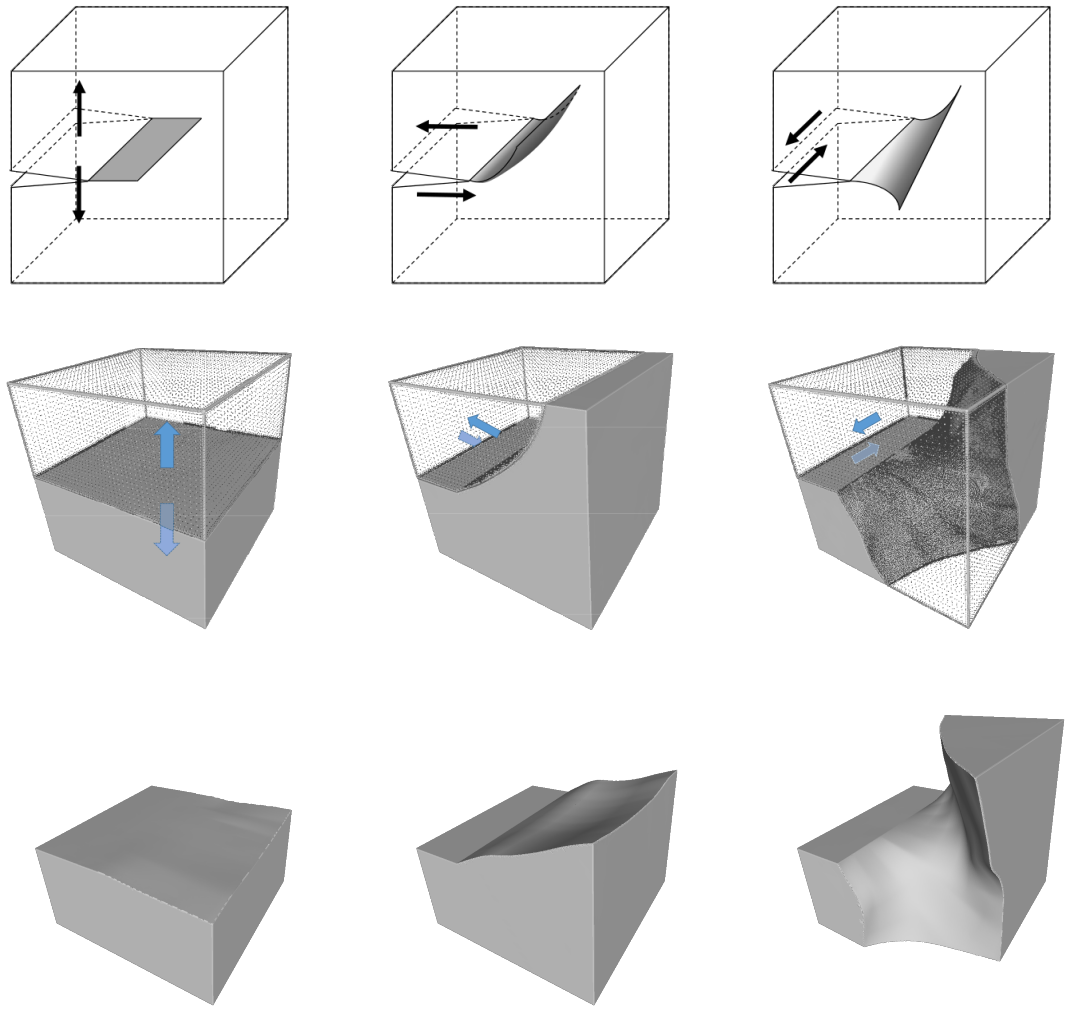
(a) (b) (c)

Figure 7: Initial forces applied on Cylinder 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 80 degree test cases respectively.



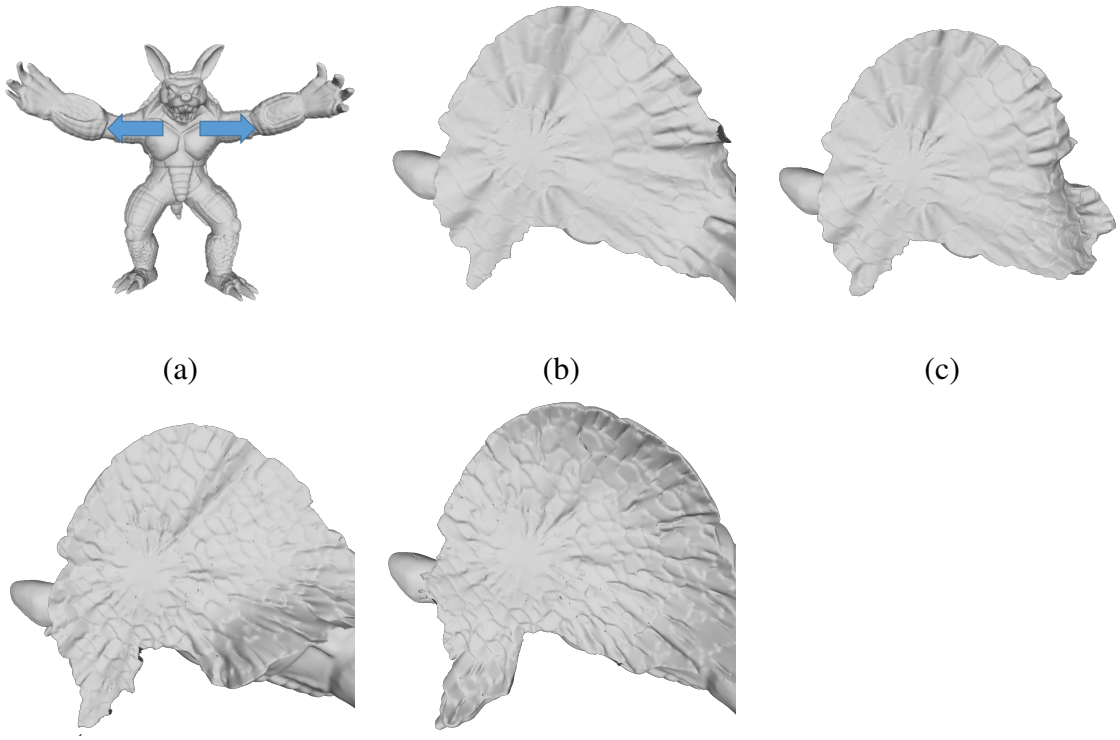
(a) (b) (c)

Figure 8: Comparisons between BEM-based simulation and our prediction method. (a) illustration of initial forces applied to the Bar model. (b) fracture surfaces of the Bar model with BEM-based simulation. (c) fracture surfaces of the Bar model with our prediction method.



mode I mode Ⅱ mode Ⅲ

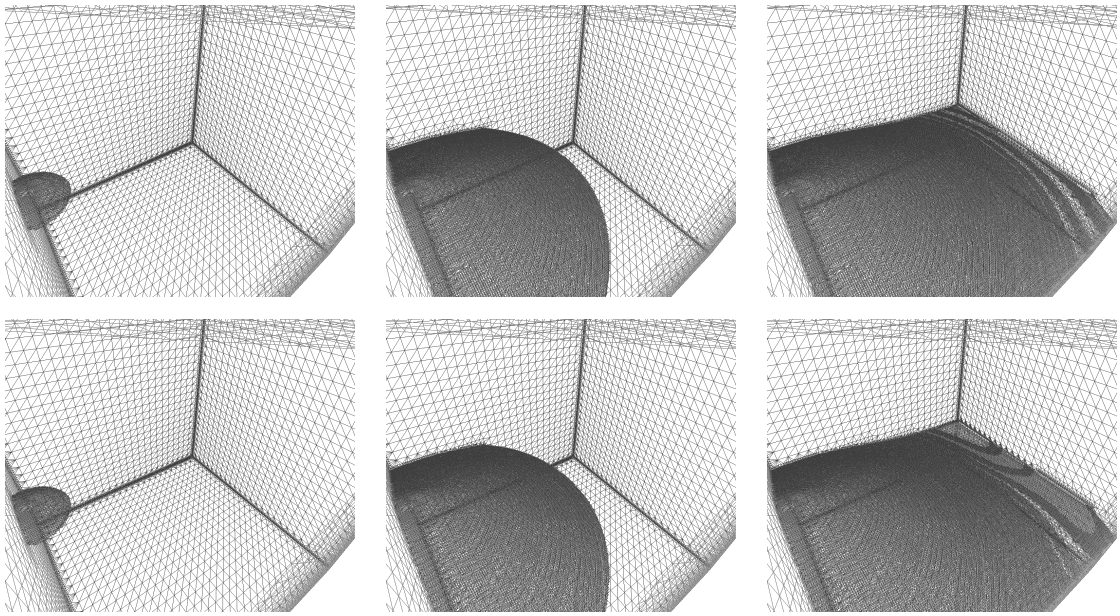
Figure 9: Comparison results between BEM-based simulation and our prediction method under three loading modes. The first row represents the definition of three loading modes (arrows) with expected crack propagation behavior (shaded), the second row represents BEM-based simulation results with initial loading modes, and the third row represents their corresponding prediction results.



(a) (b) (c)

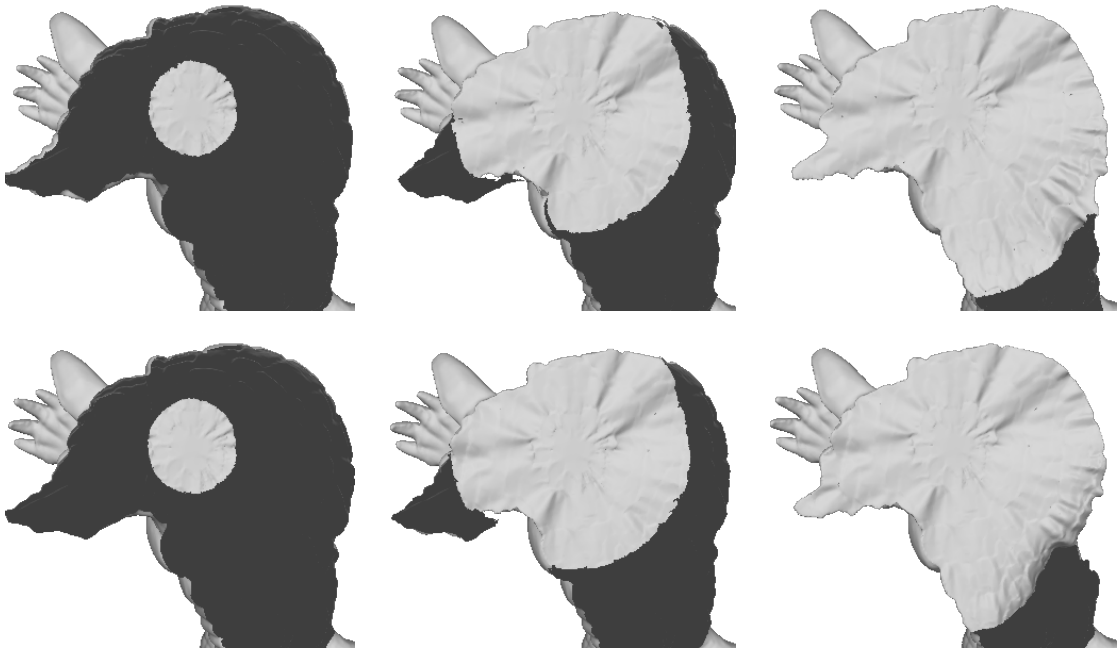
(d) (e)

Figure 10: (a) illustration of initial forces applied to models. (b) fracture surfaces of Armadillo models with BEM-based simulation. (c) fracture surfaces of Armadillo model with Armadillo database based prediction. (d) fracture surfaces of Armadillo model with Cube database based prediction. (e) fracture surfaces of Armadillo model with Cylinder database based prediction. All experiments are performed with the same initial force.



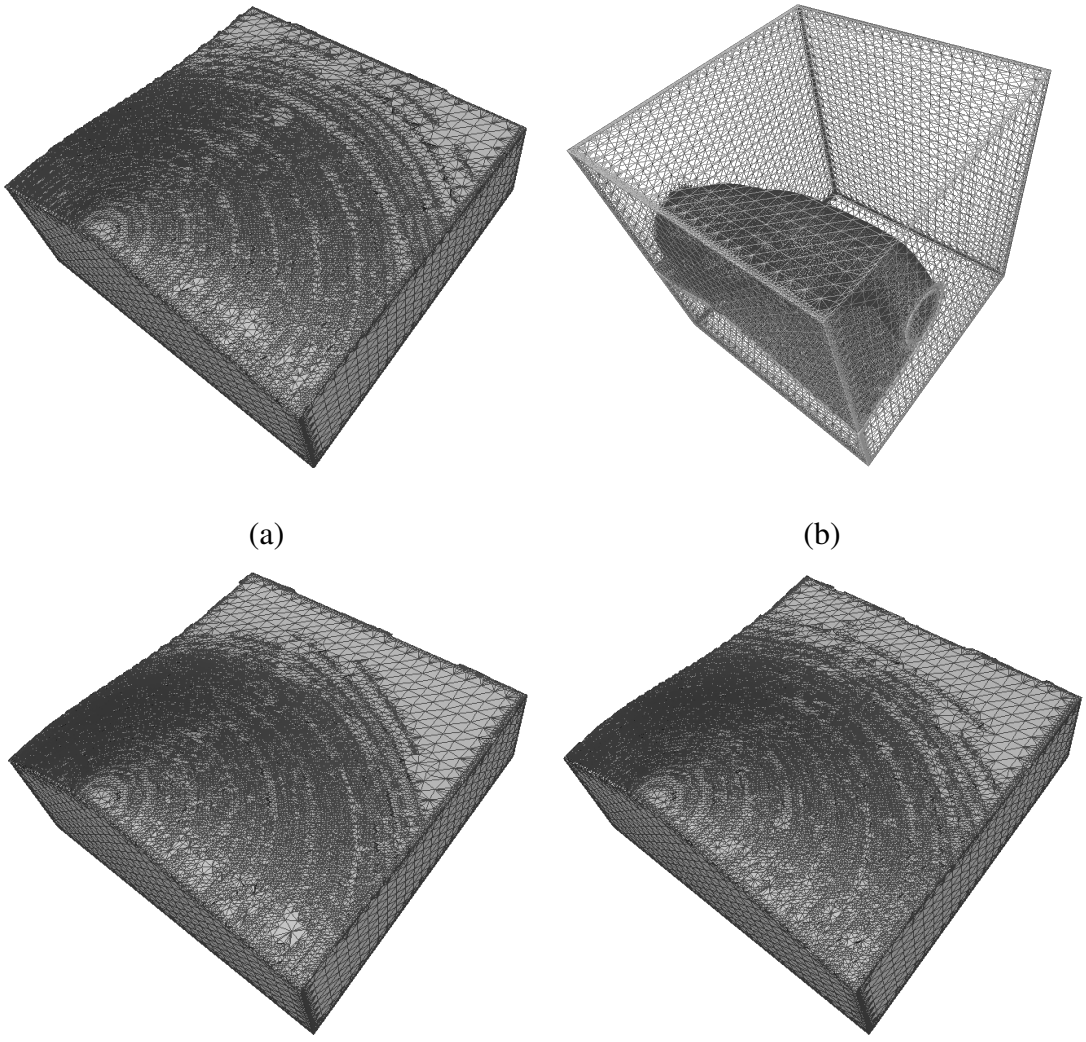
1^{st} frame 6^{th} frame 11^{th} frame

Figure 11: BEM-based simulation results on Cube model with 0 degree force direction are shown in the first row. The second row shows our data-driven results. From left to right are 1^{st}, 6^{th}, and 11^{th} frames' results respectively.



1^{st} frame 3^{rd} frame 5^{th} frame

Figure 12: BEM-based simulation results on Armadillo model are shown in the first row. The second row shows our data-driven results. From left to right are 1^{st}, 3^{th}, and 5^{th} frames' results respectively.



(a) (b)

(c) (d)

Figure 13: (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.

Type of Database	Num-sim	Num-node	(t,d)	Data generation time	Training time
RF(Cube) - 1 (used)	7502	992976	(50, 10)	11h25min16s	43min59s
RF(Cube) - 2	7502	992976	(25, 15)	11h25min16s	13h22min04s
RF(Cube) - 3	3001	411806	(50, 10)	4h23min40s	45min47s
RF(Cube) - 4	3001	411806	(25, 15)	4h23min40s	1h35min46s
RF(Cylinder) - 1 (used)	3001	492373	(50, 10)	3h30min06s	1h20min13s
RF(Cylinder) - 2	3001	492373	(25, 15)	3h30min06s	2h44min21s
RF(Armadillo)	151	12611	(10, 10)	1h37min49s	2min55s
RF(Bar)	151	71323	(50, 10)	2h13min39s	43min12s

Table 2: **Time of databases generation.** From left to right: (Num-sim) Number of times simulation executions was executed. (Num-node) Number of nodes (training samples) generated by simulations. (t, d): Parameters of regression forest training. t denotes the number of decision trees and d denotes the maximal depth of each tree. (Data generation time) Simulated time during generation of training samples. (Training time)Time for training decision trees by regression forest method.

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
	1000	1168	RF (Cube)	28.80s	0.48s	4.98s	23.34s	12	0.040s
	1000	1132	RF (Cylinder)	22.16s	0.19s	2.40s	19.57s	6	0.032s
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 3: **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.