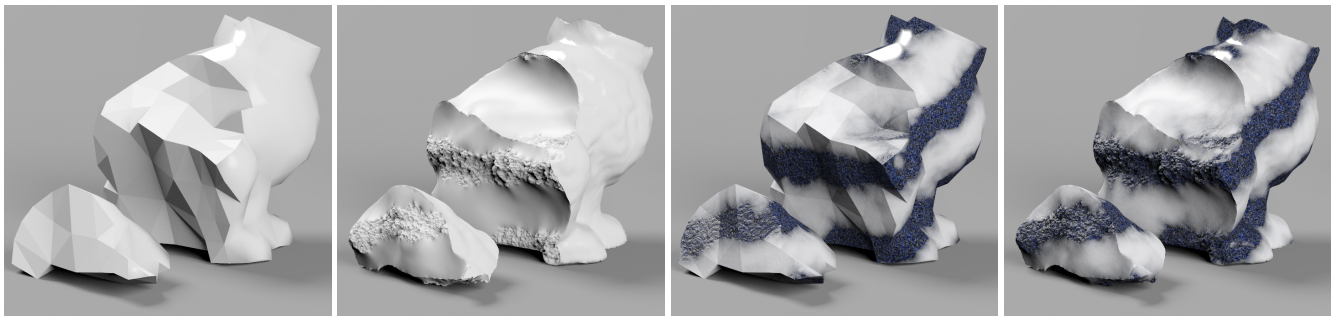
# Physics-Inspired Adaptive Fracture Refinement

Zhili Chen* Miaojun Yao* Renguo Feng* Huamin Wang*
The Ohio State University



(a) Low-resolution fracture (b) High-resolution fracture (c) Low-resolution fracture (d) High-resolution fracture

**Figure 1:** *Marble bunny. By using Perlin noise to model a material strength field and adaptive mesh refinement to improve the fracture surface, our method can generate detailed fracture effects (b) and (d) from a simulated low-resolution animation (a) and (c) in 11.7 seconds.*

## Abstract

Physically based animation of detailed fracture effects is not only computationally expensive, but also difficult to implement due to numerical instability. In this paper, we propose a physics-inspired approach to enrich low-resolution fracture animation by realistic fracture details. Given a custom-designed material strength field, we adaptively refine a coarse fracture surface into a detailed one, based on a discrete gradient descent flow. Using the new fracture surface, we then generate a high-resolution fracture animation with details on both the fracture surface and the exterior surface. Our experiment shows that this approach is simple, fast, and friendly to user design and control. It can generate realistic fracture animations within a few seconds.

**Keywords:** Fracture animation, adaptive refinement, discrete gradient flow, material strength, artistic design, animation control.

**CR Categories:** I.3.7 [Computer Graphics]: Three-Dimensional Graphics—Animation.

**Links:** DL PDF

## 1 Introduction

Physically based simulation of detailed fracture effects is a difficult problem in computer graphics. This is not only due to the computational cost that increases significantly with mesh resolution, but also due to numerical instability when the fracture process generates ill-shaped elements. Handling collisions in high-resolution fracture simulation is also complex, since many collision detection algorithms rely on proximity tests, while the created fracture surfaces are initially close to each other by default. Another issue is the lack of animation control. Fracture simulation is sensitive to physical parameters and mesh discretization. Therefore, simulated low-resolution fractures can be dramatically different from high-resolution ones, making them unsuitable for preview purposes. In recent years, the adaptive remeshing technique has demonstrated its efficiency and effectiveness in the simulation of detailed surface meshes, such as fluid surfaces [Wojtan et al. 2010; Bojsen-Hansen and Wojtan 2013], cloth [Narain et al. 2012], and thin shells [Busaryev et al. 2013; Narain et al. 2013]. Unfortunately, extending this idea to tetrahedral meshes and fracture simulation is difficult and error-prone, as Clausen and colleagues [2013] pointed out. Another issue is that the permissible time step may still be determined by the smallest element, which makes the whole system not as efficient as it should be.

Fracture details can also be generated in a non-physically based way. If the details are small, they can be simply created by displacement map. When the fracture details are large, they can be pre-defined using image textures [Mould 2005], Voronoi polygons [Raghavachary 2002], solids [Baker et al. 2011], tetrahedra [Parker and O'Brien 2009], BSP trees [Naylor et al. 1990], level sets [Su et al. 2009], or convex polyhedra [Müller et al. 2013]. Thanks to its efficiency, pre-fracturing has been widely used in the game and movie industry. Since it is not based on physics, it relies on artist to create the realism of fracture details inside of a 3D object. This can be difficult and time consuming for many artists, especially when the fractures are complex.

In this work, we propose a physics-inspired approach to enrich fracture animation effects as post process. Instead of pre-defining the fractures, we use a custom-designed material strength field to indicate where fractures are likely to happen. Given a low-resolution fracture animation, our method adaptively refines the fracture surface within a 3D object. Our contributions can be summarized into three directions:

- A Lagrangian surface evolution scheme using the discrete gradient descent flow and a modified material strength field. We formulate the gradients of the material strength field in multi-resolution, to avoid local minimum issues.

*e-mail: {chenzhi, yaom, fengr, whmin}@cse.ohio-state.edu

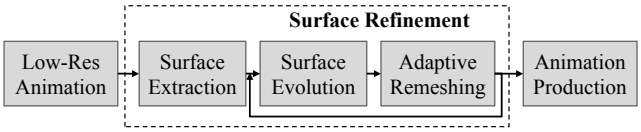



**Figure 2:** *The system pipeline.*

- An adaptive remeshing method that automatically refines the fractures according to the surface details. We further incorporate constraints, such as boundary and collision constraints, into this process.
- A high-resolution animation production approach based on deformation transfer. Using it, we can efficiently incorporate the generated high-resolution fractures into animation.

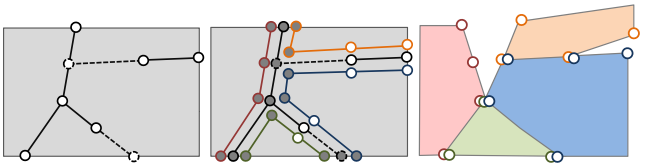
We implemented our methods as a novel fracture animation system and we tested it by a set of animation examples. Our experiment demonstrated that it can produce detailed fracture effects as shown in Figure 1. In addition, it offers a number of advantages that cannot be provided by previous systems.

- **Simple and fast**. Since our method adaptively refines the surface mesh rather the volumetric mesh, it is simple to implement and it takes only a few seconds to run, even when handling high-resolution details.
- **Artist-friendly**. We believe that providing realism in a material strength field is simpler and more intuitive than pre-fracturing, from a designer's perspective. For example, material strength is often related to material appearance and it can be modeled procedurally from solid textures.
- **Controllable**. Our system offers easy ways for user to achieve desired fracture effects in high resolution. The efficiency of our system makes it even possible for user to design and control fracture animation interactively.

## 2 Related Work

**Physically based fracture simulation.** O'Brien and his collaborators [1999; 2002] used the finite element method to simulate the fractures of brittle and ductile objects. Müller and colleagues [2001], Bao and collaborators [2007] and Zheng and James [2010] performed static analysis to simulate rigid body fractures. Parker and O'Brien [2009] and Su and colleagues [2009] improved the efficiency of fracture simulation for game applications. Molino and collaborators [2004] developed a virtual node algorithm to avoid ill-shaped elements in fracture simulation. Based on a similar idea, Sifakis and collaborators [2007] simulated highly detailed cuts using an embedded mesh. In a multi-resolution fracture approach, Müller and colleagues [2004] proposed to cut high-resolution meshes using low-resolution fractures for better surface details. Kaufmann and collaborators [2009] adopted the XFEM technique to enrich thin shell fractures with detailed cuts. Besides mesh-based fracture simulation, researchers have also studied fracture simulation in other representations, including level sets [Hegemann et al. 2013], particles [Müller 2008], polyhedra [Wicke et al. 2007; Martin et al. 2008], and point clouds [Pauly et al. 2005; Wicke et al. 2005; Steinemann et al. 2006]. Our work is particularly related to the level set fracture technique proposed by Hegemann and colleagues [2013], which also formulates the fracture as the minimal solution of an energy functional.

**Image segmentation and 3D reconstruction.** Under the variational framework, the idea of using a gradient descent flow to evolve a surface for segmentation or reconstruction purposes has been explored in computer vision [Osher and Fedkiw 2002; Jin et al. 2004; Lenkiewicz et al. 2009; Delaunoy and Prados 2011]. Our work can also be broadly considered as a 3D segmentation technique. But more importantly, our work studies how to obtain surface details through material strength field modeling and adaptive refinement, which was rarely investigated in the past.



(a) Fracture surface (b) Fracture meshes (c) Correspondences

**Figure 3:** *Surface representation. We use virtual fracture to form a connected fracture surface as shown in (a) and we use a backward remeshing step to make the mesh in the material space consistent with the mesh in every animation frame. It is then straightforward to build one-to-one mapping from the material space in (b) to the animation frame in (c).*

## 3 Overview

The pipeline of our system is shown in Figure 2. The input to this system is a low-resolution fracture animation that can be either generated by a physically based simulator, or manually created by artist. In our experiment, we used the implicit corotational FEM method [Müller et al. 2002] to simulate the dynamics and we used the separation tensor proposed by O'Brien and Hodgins [1999] to generate fracture cuts. To handle collisions, we applied the continuous collision detection approach developed by Bridson and colleagues [2002]. Since the simulator runs only in low resolution, it can robustly generate and test animation examples in real time. Given the animation input, the system extracts the low-resolution fracture surface and iteratively evolves it in the material space using a gradient flow. When we gradually deform the fracture surface, we also adaptively remesh it to better reflect material strength details. The refinement process terminates when the surface change in one iteration is sufficiently small. Finally, we can combine the high-resolution fracture surface with the high-resolution exterior surface to produce a detailed fracture animation.

## 4 Physics-Inspired Fracture Refinement

We will first present our fracture surface representation in Subsection 4.1. After that, we will discuss how to evolve it using a gradient flow, and how to adaptively refine it using mesh operations.

### 4.1 Fracture Surface Representation

To simplify our fracture surface representation, we apply two preprocessing steps to the low-resolution fracture animation. The first step is performed on the mesh in the last animation frame to eliminate incomplete fracture cuts. A fracture cut is *complete*, if it splits an object into two pieces, or *incomplete*, if it still allows an object to be connected. We propose a virtual fracture operation to extend incomplete cuts to complete ones, by propagating the cuts further in their original separation directions and computing their intersections with other surfaces. Doing this may introduce more disconnected pieces due to the self intersection of the newly generated virtual fracture cuts, but since they are not really separated in the final animation, they are not supposed to cause splitting artifacts. The result is a connected non-manifold fracture surface that separates the object into a set of connected components, as Figure 3a shows.

(a) Strength Field (b) Stress Field (c) Modified Strg. Field

**Figure 4:** *Modified material strength field. By subtracting an estimated stress field in (b) from a procedurally generated strength field in (a), we generate a modified material strength field in (c) to model the fractures of a collapsing brick wall.*

Our second step is to address mesh inconsistency when the original mesh was remeshed by the fracture simulation technique proposed by O'Brien and Hodgins [1999]. Our idea is to perform a backward remeshing operation that uses the processed mesh in the last frame to rebuild the meshes in previous frames. By doing this, we make sure that the meshes in all of the frames have the same topology. Intuitively, this means the object has been separated topologically into multiple components, even before fracture events happen.

We use the same representation in the material space to represent the low-resolution fracture surface as Figure 3b shows. Since the meshes in the material space and in the animation frames have the same topology, we can easily establish one-to-one mapping from a vertex in the material space to a vertex in a frame. To help recognize those vertices belonging to the same original fracture surface vertex, we use a list to link them together. We also label out the special edges and their vertices, if they are on the fracture surface boundary, or shared by more than two triangles in the non-manifold fracture surface, as those gray dots shown in Figure 3b. Finally, we use flood fill to assign each fractured component and its mesh with a color as shown in Figure 3c. We will use this color information to handle deformation transfer in Section 5.

### 4.2 Modified Material Strength Field

In materials science, the strength of a material describes its ability to withstand a given stress. So a fracture is formed within an object, when its loaded stress is beyond its material strength. If the stress is uniform, a fracture cut must be located where the strength is the weakest. The problem is that the stress is unlikely to be uniformly distributed in the real world. Our idea is to approximate stress variation using the low-resolution fracture surface, and then model a modified material strength field for surface evolution afterwards.

Assuming that the material strength is isotropic, we represent it in the material space of an object using a 3D scalar field $\zeta(\mathbf{x})$. This field can be derived procedurally using solid textures, or synthesized from image exemplars [Kopf et al. 2007], or even custom designed by artist. Compared with the material strength field, the stress field is much more difficult to obtain. This is not only because the stress field varies dynamically over time, but also because it depends heavily on the fracture, which is supposed to be updated by our system. To solve this problem, we assume that the low-resolution fracture surface indicates the high stress region well. So we can estimate the stress field $\xi(\mathbf{x})$ using the Euclidean distance from any point $\mathbf{x}$ to the low-resolution surface in the material space. After we get $\zeta(\mathbf{x})$ and $\xi(\mathbf{x})$, we subtract $\xi(\mathbf{x})$ from $\zeta(\mathbf{x})$ to generate a modified material strength field $\psi(\mathbf{x})$. Intuitively, $\psi(\mathbf{x})$ indicates the possible region where the stress is high and the material strength is weak. Figure 4 shows one example of a modified material strength field, in which the low-resolution fracture surface is a straight line.

The use of this modified strength field allows convenient fracture animation control in two ways. First of all, an artist can adjust the definition of each function. For example, by making $\xi(\mathbf{x})$ more important in $\psi(\mathbf{x})$, the resulting fracture surface will become closer to the original surface. Secondly, an artist can directly modify the low-resolution fracture surface in the material space. The estimated stress field $\xi(\mathbf{x})$ prevents the fracture surface from traveling too far away during surface evolution. So the low-resolution fracture animation can be effectively used for design and preview purposes, before generating high-resolution results.

### 4.3 The Gradient Flow and Surface Evolution

Given the modified strength field $\psi(\mathbf{x})$, our next goal is find a new fracture surface $\mathbf{S}$ that minimizes the following energy functional:

$$\mathcal{E}(\mathbf{S}) = \int_{\mathbf{S}} \psi(x) ds. \tag{1}$$

Using the generic formulation of the discrete gradient descent flow given by Delaunoy and Prados [2011], we can compute the evolution velocity of vertex $i$ as:

$$\frac{d\mathbf{x}_i}{dt} = -\frac{1}{A_i} \sum_{j \in N_i} \left( \int_{\mathbf{S}_j} \nabla_{\mathbf{x}_i} \psi(\mathbf{x}) \phi_i(\mathbf{x}) ds - \frac{\mathbf{e}^i_j \times \mathbf{n}_j}{2A_j} \int_{\mathbf{S}_j} \psi(\mathbf{x}) ds \right), \tag{2}$$

in which $A_i$ is the 1-ring surface area of vertex $i$, $\phi_i(\mathbf{x})$ is the linear basis function, $A_j$ and $\mathbf{n}_j$ are the area and the normal of triangle $j$ in vertex $i$'s neighborhood, and $\mathbf{e}^i_j$ is an edge of triangle $j$ opposing to vertex $i$. To avoid the complexity of evaluating the first integral in Equation 2, we assume[1] that $\nabla_{\mathbf{x}_i} \psi(\mathbf{x})$ is equivalent to $\nabla \psi(\mathbf{x}_i)$ within the triangle surface. By approximating $\psi(\mathbf{x})$ in the triangle using linear interpolation, we can reformulate Equation 2 into:

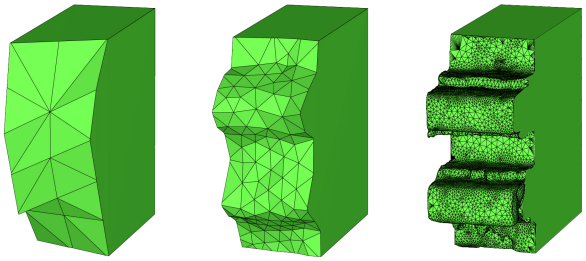
$$\frac{d\mathbf{x}_i}{dt} = -\frac{1}{A_i} \sum_{j \in N_i} \left( \frac{1}{3} A_j \nabla \psi(\mathbf{x}_i) - \frac{\mathbf{e}^i_j \times \mathbf{n}_j}{2A_j} \sum_{k \in T_j} \psi(\mathbf{x}_k) \right), \tag{3}$$

in which $k$ is a vertex of triangle $j$. We note that when $\psi(\mathbf{x}) \equiv 1$, $\mathcal{E}(\mathbf{S})$ in Equation 1 becomes the surface energy and Equation 3 is identical to the surface-based mean curvature flow proposed by Meyer and collaborators [2002]. Meanwhile, if the surface is locally planar at vertex $i$ and $\nabla \phi(\mathbf{x})$ is constant and parallel to the surface, it can be proved that $\frac{d\mathbf{x}_i}{dt} = 0$. In other words, the vertices will not be clustered, due to material strength variations on the surface.

Since $i$ is a vertex of the non-manifold surface in Equation 3, to enumerate all of its adjacent triangles, we use the linked list presented in Subsection 4.1 to find its corresponding vertices and their adjacent triangles in our surface representation. We then compute its new position and use the linked list again to update the positions of all corresponding vertices. We note that the discrete gradient flow in Equation 3 works well for non-manifold fracture surfaces in our case, even though it was initially developed for manifold surfaces.

A critical question here is how we can compute the gradient of $\psi(\mathbf{x})$. In our system, we use a uniform grid to present the strength field $\psi(\mathbf{x})$ and we use finite differencing to calculate its gradient. But if the gradient kernel is not large enough, this method can easily suffer from the local minimum issue. Inspired by feature detection and image segmentation techniques (such as SIFT [Lowe 2004]), we propose to use a Gaussian smoothing filter to build a material strength field pyramid during the initialization step. After that, we compute the gradient of each vertex as the average of its gradients at all pyramid levels. We typically use 4 to 6 pyramid levels in our experiment. Doing this effectively avoids the local minimum issue.

[1] Mathematically, this is similar to replacing the "mass" matrix by the vertex area $A_i$, as suggested by Eckstein and colleagues [2007].



(a) Original (b) After two iterations (c) After 16 iterations

**Figure 5:** *Fracture surface mesh during the refinement process.*

### 4.4 Adaptive Fracture Remeshing

To make the fracture surface more accurately reflect the underlying details in the material strength field, we adaptively remesh it after it is evolved in each iteration. A typical adaptive remeshing technique proposed by Narain and colleagues [2012; 2013] is to perform edge split and edge merge. While this technique can handle detailed surface wrinkles, it is not very suitable in our system, when the details are defined over the surface area. So instead, we propose to randomly select a sample within a triangle and compute the new position of this point using the evolution function in Equation 3, in which case its neighborhood is the whole triangle. If it moves too far away from the triangle after one iteration, we add it as a new vertex and subdivide the triangle into three new triangles. To avoid unnecessary computation, we discard a sample point if it is too close to a triangle vertex. To refine the fracture surface boundary, we select samples on the boundary edges instead, if a triangle is on the boundary. For the same reason, we select samples on an edge if it was originally shared by more than two triangles in the non-manifold fracture surface. Although similar coplanar criteria can be used to remove a vertex and re-triangulate its neighborhood, we do not think it is necessary, since the surface is supposed to obtain more details during its evolution. Once we create a new vertex and subdivide a triangle for one connected component, we use the linked list described in Subsection 4.1 to update the meshes of other components accordingly. This ensures that the underlying non-manifold fracture surface is still consistently represented.

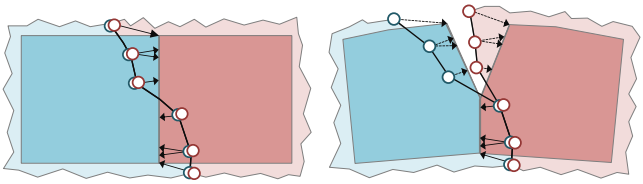
After we remesh the fracture surface, we perform three additional steps to improve it. Similar to the remeshing step, these steps are performed once for all object components using the linked list. Figure 5 shows the results of the overall mesh refinement process of the brick wall example.

**Edge flip.** We apply the edge flip method on an edge to reduce slim triangles caused by surface evolution or remeshing, unless the edge cannot be flipped (i.e., does not have exactly two triangle neighbors on the original non-manifold surface).

**Boundary constraint.** The vertices on the fracture surface boundary must stay on the exterior surface of the object. To prevent them from escaping the exterior surface, we project them back onto a user-defined high-resolution exterior surface after each iteration.

**Collision constraints.** We use the continuous collision detection approach [Bridson et al. 2002] to detect self collisions of the fracture surface. After we find each colliding vertex-triangle or edge-edge pair, we simply stop their vertex motions to prevent them from penetrating into each other.

A newly added vertex does not have its correspondence in animation, which can become a problem when we calculate the deformation for every fracture surface vertex during our high-resolution animation production in Section 5. To solve this problem, we associate each new vertex with the three triangle indices and the barycentric weights, from which it was created. Since we perform the adaptive refinement step multiple times, we can intuitively consider this data structure as a tree, where the root is a new vertex and the leaves are the old vertices in the low-resolution mesh. By doing linear interpolation, we can use it to compute various quantities for every vertex on the refined mesh, such as deformation and fracture time.



(a) Material space (b) Animation space

**Figure 6:** *Fracture surface generation. By using the tree structure and linear interpolation to estimate the deformation of each high-resolution fracture vertex in the material space, we can transfer the high-resolution fracture surface from the material space to the animation space in each frame.*

## 5 High-Resolution Animation Production

After we obtain a detailed fracture surface in Section 4, we can now generate high-resolution fracture animation. Our basic idea here is to replace the low-resolution meshes in each frame by the high-resolution meshes in the material space using deformation transfer.

**Fracture surface generation.** To replace the low-resolution fracture surface by the high-resolution one in each animation frame, we first compute the deformation of each low-resolution fracture vertex. If the low-resolution fracture animation was simulated as in our experiment, we can calculate the deformation of each vertex by interpolating the deformation of its adjacent tetrahedra. If the fracture surface and the animation was created by artist instead, we can still use the approach proposed by Sumner and Popović [2004] to calculate vertex deformation from its adjacent triangles. After that, we use the tree data structure in Subsection 4.4 to propagate the deformation from low-resolution mesh vertices to high-resolution ones in each animation frame. Specifically, the tree structure indicates that each vertex on the high-resolution fracture surface can be considered as a combination of the vertices on the low-resolution surface, as Figure 6a shows in the material space. Using the associated interpolation weights, we can interpolate vertex deformation and then transfer the high-resolution fracture surface from the material space to the animation space. This implies that the object is always disconnected topologically over the whole animation sequence. However, the actual separation events are determined by the underlying deformation. If the components have different deformations as in Figure 6b, they will be disconnected. Otherwise, they will be connected until they receive different deformations.

**Exterior surface generation.** After we replace the fracture surface, we also need to replace the exterior surface in animation by a user-defined high-resolution one. Our first step is to calculate the intersection between the high-resolution exterior surface $\mathbf{B}$ and the high-resolution fracture surface $\mathbf{S}$. According to Subsection 4.4, each vertex on the high-resolution fracture surface boundary $\partial\mathbf{S}$ has already been projected onto $\mathbf{B}$. So we apply the surface traversal technique (such as in [Chen et al. 2013]) to calculate the geodesic path of $\partial\mathbf{S}$ and find the intersection points between $\partial\mathbf{S}$ and $\mathbf{B}$, as the small dots shown in as Figure 7. We then use the intersection points to remesh the boundary triangles of both $\mathbf{B}$ and $\mathbf{S}$, and we use the geodesic path to split $\mathbf{B}$ into multiple patches, each of which

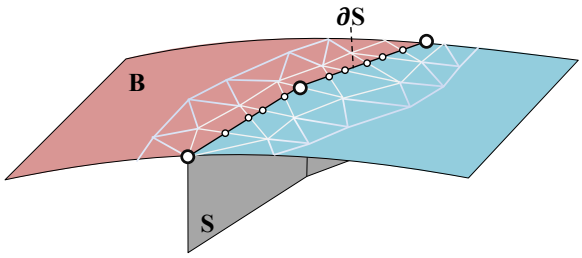
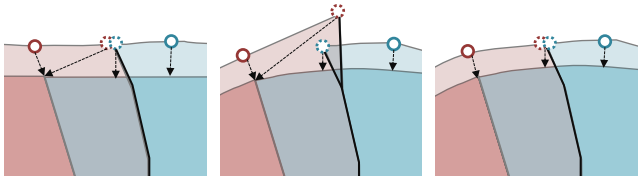



**Figure 7:** *Exterior surface generation. We use surface traversal to calculate the intersection between the fracture surface* **S** *and the exterior surface* **B**. *After that, we remesh both surfaces and we merge them to obtain a watertight mesh for each component.*



(a) Material space (b) Early fracture (c) Our solution

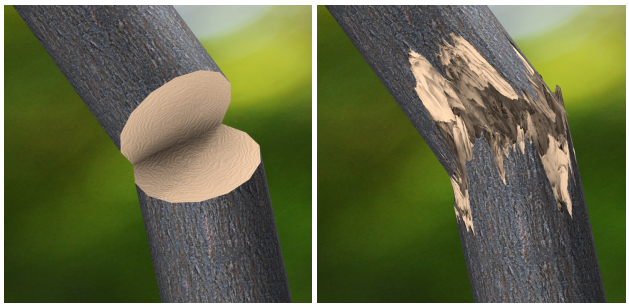
**Figure 8:** *Early fracture. When boundary vertices are mapped to different points on the low-resolution surface, they will receive different deformations and cause an early fracture problem. By considering the actual fracture time, our solution fixes this problem.*

represents the exterior surface of one component. After that, we merge **S** and each patch to form a watertight high-resolution mesh for every component.

For animation production, we also transfer the high-resolution exterior surface from the material space to the animation space. To obtain the deformation of each high-resolution exterior vertex, we simply find the nearest low-resolution surface location that belongs to the same component, as shown in Figure 8a. We then transfer the deformation of that nearest location.

**Discussion.** When the object undergoes large deformation and the refinement process causes large difference between the low-resolution fracture and the high-resolution fracture, an early fracture problem may appear as Figure 8b shows. In this example, high-resolution boundary vertices corresponding to the same location in the material space are mapped to different low-resolution locations, since they belong to different components. As a result, the vertices receive different deformations and the components become disconnected, before the actual fracture happens. To solve this problem, we compute the fracture time of each vertex according to the low-resolution animation, using the tree data structure in Subsection 4.4. We then use the same nearest low-resolution location to calculate the deformation for both boundary vertices regardless of their component identities as shown in Figure 8c, until the fracture time is reached. Doing this will eliminate the early fracture issue, but it may cause a sudden change on the object surface when the fracture begins. Fortunately, if the fracture is formed gradually, this change is small; if the fracture is formed rapidly, it is hardly noticeable, as our experiment shows.

A more problematic issue is self collisions due to the changed fracture details. Unfortunately, the only viable solution we know is to treat our result as pre-defined fractures and re-simulate the whole animation in high resolution, which will be computationally expensive. We ignore such a problem in our current system at this time.



(a) Original result (b) Our result

**Figure 9:** *Tree branch. Using a simple material strength field, our method can generate detailed fracture patterns on this tree branch.*

## 6 Results and Discussions

(Please see the supplementary video for more animation examples.) We test the performance of our algorithms using a single core of a 3.4GHz Intel Core i7-2600 processor. The material strength fields in all of the four examples are modeled by procedural functions. We use a regular grid to sample each procedural function and build a material strength pyramid. This initialization step typically takes about 10 to 30 seconds to run and it is done only once. The mesh refinement process in our system requires most of the computational cost, which is fortunately independent of the animation length specified by the low-resolution animation input. After mesh refinement, high-quality animation can be produced in real time as proposed in Section 5.

**Marble bunny.** In this example, we assign the bunny with a solid marble texture as Figure 1 shows. The blue region of this texture, modeled by high-frequency Perlin noise, contains fine and hard grains; the white region of this texture, modeled by low-frequency Perlin noise, is smooth and soft. We sample this texture using a $50 \times 50 \times 50$ grid. Our system refines a low-resolution fracture surface with 256 triangles to a high-resolution surface with 174K triangles in 11.7 seconds.

**Tree branch.** We model a broken tree branch as a cylinder separated by an incomplete fracture cut in the middle, as Figure 9 shows. We procedurally define the strength of this cylinder, such that it is soft inside but hard outside. We also add elongated Perlin noise to make the fracture surface favor the cylinder axis direction. We use a $32 \times 32 \times 64$ grid and we spend 5.0 seconds to generate a high-resolution fracture surface containing 123K triangles, from a low-resolution one containing 60 triangles.

**Collapsing wall.** A brick wall collapses due to a wrecking ball in this example. The material strength of this wall is strong in both the bricks and the mortar, but weak on the boundary. As a result, the fracture surface is likely to follow the brick-mortar boundary as Figure 10 shows. Here the material strength field is represented by a $150 \times 50 \times 10$ grid. It takes 13.6 seconds to refine the fracture surface from 204 triangles to 208K triangles.

**Plastic clay.** This example (in Figure 11) demonstrates the use of our system in handling fracture cuts caused by large plastic and twisting deformation, as long as the deformation is covered by low-resolution simulation. Here we use a $32 \times 64 \times 64$ grid to represent the material strength field. The low-resolution fracture surface contains 510 triangles and the refined one contains 40K triangles. The refinement process takes 3 seconds to run.

**Jell-O.** Our final example is a bouncy Jell-O stuffed with fruit cubes in different colors, as shown in Figure 12. It shows that our

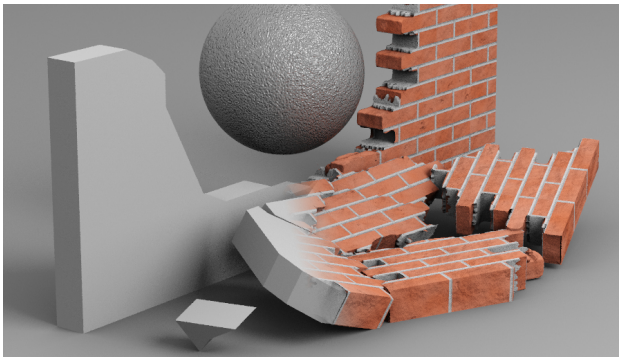
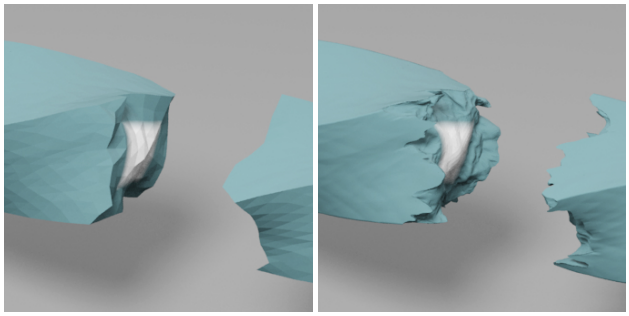


**Figure 10:** *Collapsing wall. In less than 14 seconds, our method generates highly detailed fracture surfaces for this brick wall.*



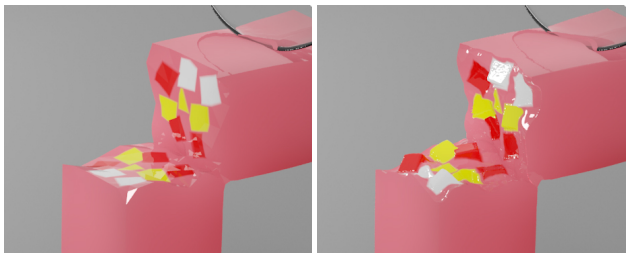
(a) Original result (b) Our result

**Figure 11:** *Twisted plastic clay. Our method generates detailed fractures under large twisting and plastic deformation, as long as it is covered by the low-resolution animation input.*

system can also handle gradual fracture effects in large elastic deformation. We use a $64 \times 64 \times 32$ grid to represent the material strength field in this example. The low-resolution fracture surface contains 488 triangles and the refined one contains 32K triangles. The refinement process runs for 3 seconds.

Our preliminary test shows that it takes more than 30 minutes for our physically based simulator to generate one frame in the same resolution as ours, which is orders-of-magnitude slower than our method. Since it is too time consuming and our simulator may suffer from numerical instability in high resolution, we do not simulate high-resolution results for comparison purposes. We would like to emphasize that it is also difficult to obtain desired, controllable fracture effects from high-resolution fracture simulation, but it is easy to achieve by our method.

**Limitations.** While our method can produce realistic fracture surfaces in high resolution, it is not supposed to be physically accurate. Our method considers only the heterogeneity of the material strength, and it is not clear to us whether it can be extended to handle anisotropic material strength. When our method evolves and refines the fracture surface, it maintains the surface topology so that it cannot create novel fracture cuts. Besides, if the refined fracture is too far away from original fracture, the high-resolution animation cannot fully transfer nonlinear elastic or plastic deformation near the fracture cut. Our method refines only the fracture surface details, but not the fracture time. In other words, the fracture formation in high-resolution animation may still not be smooth, due to the low-resolution animation. Currently, we require the fracture surface meshes to move consistently as a single non-manifold surface. But we should allow them to move separately in the future, to handle objects with interior holes. Finally, handling collisions



(a) Original result (b) Our result

**Figure 12:** *Jell-O with fruit cubes. Our method can also handle fractures in large elastic deformation, as shown in this example.*

in high resolution is computationally expensive and we do not consider it so far. It can be easily added, if it is necessary.

## 7 Conclusions and Future Work

In conclusion, we proposed an adaptive refinement approach to produce high-resolution fracture details in animation. Our experiment demonstrated that these details can be efficiently and easily generated without physically based simulation in high resolution. This approach can be used as a powerful tool for designers and artists to create high-quality fracture animation effects with easy control.

Looking into the future, our immediate plan is to make our system more efficient, by improving our remeshing scheme to reduce unnecessary triangles. We are also interested in other ways to produce high-resolution animation, such as re-simulation using new embedded meshes as proposed by Sifakis and collaborators [2007]. Finally, we plan to study incremental fracture refinement without using the whole low-resolution animation sequence. The resulting technique will be useful for interactive applications, such as games.

## 8 Acknowledgment

The authors would like to thank Nvidia and Adobe for their funding and equipment support. This work is also funded by the open project program of the State Key Lab of CAD&CG at Zhejiang University.

## References

Baker, M., Carlson, M., Coumans, E., Criswell, B., Harada, T., Knight, P., and Zafar, N. B. 2011. Destruction and dynamic artist tools for film and game production. In *ACM SIGGRAPH 2011 course notes*.

Bao, Z., Hong, J.-M., Teran, J., and Fedkiw, R. 2007. Fracturing rigid materials. *IEEE Transactions on Visualization and Computer Graphics 13*, 2 (Mar.), 370–378.

Bojsen-Hansen, M., and Wojtan, C. 2013. Liquid surface tracking with error compensation. *ACM Trans. Graph. (SIGGRAPH) 32*, 4 (July), 68:1–68:13.

Bridson, R., Fedkiw, R., and Anderson, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph. (SIGGRAPH) 21*, 3 (July), 594–603.

Busaryev, O., Dey, T. K., and Wang, H. 2013. Adaptive fracture simulation of multi-layered thin plates. *ACM Trans. Graph. (SIGGRAPH) 32*, 4 (July), 52:1–52:6.

CHEN, Z., FENG, R., AND WANG, H. 2013. Modeling friction and air effects between cloth and deformable bodies. *ACM Trans. Graph. (SIGGRAPH) 32*, 4 (July), 88:1–88:8.

CLAUSEN, P., WICKE, M., SHEWCHUK, J. R., AND O'BRIEN, J. F. 2013. Simulating liquids and solid-liquid interactions with lagrangian meshes. *ACM Trans. Graph. 32*, 2 (Apr.), 17:1–15.

DELAUNOY, A., AND PRADOS, E. 2011. Gradient flows for optimizing triangular mesh-based surfaces: Applications to 3D reconstruction problems dealing with visibility. *International Journal of Computer Vision 95*, 2, 100–123.

ECKSTEIN, I., PONS, J.-P., TONG, Y., KUO, C.-C. J., AND DESBRUN, M. 2007. Generalized surface flows for mesh processing. In *Proc. of SGP*, 183–192.

HEGEMANN, J., JIANG, C., SCHROEDER, C., AND TERAN, J. M. 2013. A level set method for ductile fracture. In *Proc. of SCA*, 193–201.

JIN, H., YEZZI, A. J., AND SOATTO, S. 2004. Region-based segmentation on evolving surfaces with application to 3D reconstruction of shape and piecewise constant radiance. In *ECCV*, 114–125.

KAUFMANN, P., MARTIN, S., BOTSCH, M., GRINSPUN, E., AND GROSS, M. 2009. Enrichment textures for detailed cutting of shells. *ACM Trans. Graph. (SIGGRAPH) 28*, 3 (July), 50:1–50:10.

KOPF, J., FU, C.-W., COHEN-OR, D., DEUSSEN, O., LISCHINSKI, D., AND WONG, T.-T. 2007. Solid texture synthesis from 2D exemplars. *ACM Trans. Graph. (SIGGRAPH) 26*, 3 (July).

LENKIEWICZ, P., PEREIRA, M., FREIRE, M. M., AND FERNANDES, J. 2009. The whole mesh deformation model for 2D and 3D image segmentation. In *ICIP*, IEEE, 4045–4048.

LOWE, D. G. 2004. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision 60*, 2 (Nov.), 91–110.

MARTIN, S., KAUFMANN, P., BOTSCH, M., WICKE, M., AND GROSS, M. 2008. Polyhedral finite elements using harmonic basis functions. In *Proc. of SGP*, 1521–1529.

MEYER, M., DESBRUN, M., SCHRODER, P., AND BARR, A. H. 2002. Discrete differential-geometry operators for triangulated 2-manifolds. In *VisMath*, Springer-Verlag, 35–57.

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

MOULD, D. 2005. Image-guided fracture. In *Proceedings of Graphics Interface 2005*, GI '05, 219–226.

MÜLLER, M., AND GROSS, M. 2004. Interactive virtual materials. In *Proceedings of Graphics Interface 2004*, GI '04, 239–246.

MÜLLER, M., MCMILLAN, L., DORSEY, J., AND JAGNOW, R. 2001. Real-time simulation of deformation and fracture of stiff materials. In *Proc. of SCA*, 113–124.

MÜLLER, M., DORSEY, J., MCMILLAN, L., JAGNOW, R., AND CUTLER, B. 2002. Stable real-time deformations. In *Proc. of SCA*, 49–54.

MÜLLER, M., CHENTANEZ, N., AND KIM, T.-Y. 2013. Real time dynamic fracture with volumetric approximate convex decompositions. *ACM Trans. Graph. (SIGGRAPH) 32*, 4 (July), 115:1–115:10.

MÜLLER, M. 2008. Hierarchical position based dynamics. In *VRIPHYS*, 1–10.

NARAIN, R., SAMII, A., AND O'BRIEN, J. F. 2012. Adaptive anisotropic remeshing for cloth simulation. *ACM Trans. Graph. (SIGGRAPH Asia) 31*, 6 (Nov.), 152:1–152:10.

NARAIN, R., PFAFF, T., AND O'BRIEN, J. F. 2013. Folding and crumpling adaptive sheets. *ACM Trans. Graph. (SIGGRAPH) 32*, 4 (July), 51:1–51:8.

NAYLOR, B., AMANATIDES, J., AND THIBAULT, W. 1990. Merging BSP trees yields polyhedral set operations. *SIGGRAPH Comput. Graph. 24*, 4 (Sept.), 115–124.

O'BRIEN, J. F., AND HODGINS, J. K. 1999. Graphical modeling and animation of brittle fracture. In *Proc. of SIGGRAPH 98*, Annual Conference Series, 137–146.

O'BRIEN, J. F., BARGTEIL, A. W., AND HODGINS, J. K. 2002. Graphical modeling and animation of ductile fracture. *ACM Trans. Graph. (SIGGRAPH) 21*, 3 (July), 291–294.

OSHER, S. J., AND FEDKIW, R. P. 2002. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag.

PARKER, E. G., AND O'BRIEN, J. F. 2009. Real-time deformation and fracture in a game environment. In *Proc. of SCA*, 165–175.

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

RAGHAVACHARY, S. 2002. Fracture generation on polygonal meshes using voronoi polygons. In *ACM SIGGRAPH 2002 Conference Abstracts and Applications*, SIGGRAPH '02, 187–187.

SIFAKIS, E., DER, K. G., AND FEDKIW, R. 2007. Arbitrary cutting of deformable tetrahedralized objects. In *Proc. of SCA*, 73–80.

STEINEMANN, D., OTADUY, M. A., AND GROSS, M. 2006. Fast arbitrary splitting of deforming objects. In *Proc. of SCA*, 63–72.

SU, J., SCHROEDER, C., AND FEDKIW, R. 2009. Energy stability and fracture for frame rate rigid body simulations. In *Proc. of SCA*, 155–164.

SUMNER, R. W., AND POPOVIĆ, J. 2004. Deformation transfer for triangle meshes. *ACM Trans. Graph. (SIGGRAPH) 23*, 3 (Aug.), 399–405.

WICKE, M., STEINEMANN, D., AND GROSS, M. H. 2005. Efficient animation of point-sampled thin shells. *Computer Graphics Forum (Eurographics) 24*, 3, 667–676.

WICKE, M., BOTSCH, M., AND GROSS, M. 2007. A finite element method on convex polyhedra. *Computer Graphics Forum (Eurographics) 26*, 3, 355–364.

WOJTAN, C., THÜREY, N., GROSS, M., AND TURK, G. 2010. Physics-inspired topology changes for thin fluid features. *ACM Trans. Graph. (SIGGRAPH) 29*, 4 (July), 50:1–50:8.

ZHENG, C., AND JAMES, D. L. 2010. Rigid-body fracture sound with precomputed soundbanks. *ACM Trans. Graph. (SIGGRAPH) 29*, 4 (July), 69:1–69:13.