Computer Methods and Programs in Biomedicine 219 (2022) 106749

Contents lists available at ScienceDirect

Computer Methods and Programs in Biomedicine

journal homepage: www.elsevier.com/locate/cmpb

Versatile cutting fracture evolution modeling for deformable object cutting simulation

Sirui He[a,b,1,1], Yinling Qian[b,1], Xin Zhu[a,b], Xiangyun Liao[b], Pheng-Ann Heng[c,b], Ziliang Feng[a], Qiong Wang[b,*]

[a] College of Computer Science, Sichuan University, China
[b] Guangdong Provincial Key Laboratory of Computer Vision and Virtual Reality Technology, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, China
[c] Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong

ARTICLE INFO

Article history:
Received 22 October 2021
Revised 9 February 2022
Accepted 10 March 2022

Keywords:
Virtual cutting simulation
Cutting fracture modeling
Griffith's energy minimization

ABSTRACT

Background and Objectives: Soft body cutting simulation is the core module of virtual surgical training systems. By making full use of the powerful computing resources of modern computers, the existing methods have already met the needs of real-time interaction. However, there is still a lack of high realism. The main reason is that most current methods follows the "Intersection-IS-Fracture" mode, namely cutting fracture occurs as long as the cutting blade intersects with the object. To model real-life cutting phenomenon considering deformable objects' fracture resistance, this paper presents a highly realistic virtual cutting simulation algorithm by introducing an energy-based cutting fracture evolution model.

Methods: We design the framework based on the co-rotational linear FEM model to support large deformations of soft objects and also adopt the composite finite element method (CFEM) to balance between simulation accuracy and efficiency. Then, a cutting plane constrained Griffth's energy minimization scheme is proposed to determine when and how to generate a new cut. Moreover, to provide the contact effect before the fracture occurs, we design a material-aware adaptation scheme that can guarantee indentation consistent with the cutting tool blade and visually plausible indentation-induced deformation to avoiding large computational effort.

Results and conclusion: The experimental results demonstrate that the proposed algorithm is feasible for generating highly realistic cutting simulation results of different objects with various materials and geometrical characteristics while introducing a negligible computational cost. Besides, for different blade shapes, the proposed algorithm can produce highly consistent indentation and fracture. Qualitative evaluation and performance analysis indicate the versatility of the proposed algorithm.

1. Introduction

As the core module of virtual surgical training systems, deformable object cutting simulation has been an important research topic for nearly two decades. The existing methods mainly focus on solving the three major tasks for virtual cutting simulation, namely the simulation of the deformable body, the detection, and handling of collision, and the incorporation of cuts into the computational model [1]. According to the underlying deformation model, cutting simulation methods can be divided into two categories [2]: mesh-based methods [3–6], mesh-less methods [7,8].

The most common mesh-based method is the finite element method (FEM), which is the majority of simulation cutting of deformable models in the early years. Nielsen proposed the pioneering work which directly removes elements along the cutting path directly [9]. Even this method is straightforward, it not only led to jagging artifacts but also violates the law of conservation of mass. Various approaches are proposed to eliminate the jagging artifacts, such as splitting elements touched by the cutting tool [10] or embedding an accurate surface into the original shapes by duplicating elements. However, these methods can not accurately accommodate intricate cuts with a number of elements. To solve this problem, Bielser proposed an element refinement method [11]. It decomposes the tetrahedron by generating a vertex on each edge and each triangle face when cutting. Further, Bielser improvements for the element refinement method by using the split method [12] and

* Corresponding author.
E-mail address: wangqiong@siat.ac.cn (Q. Wang).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.cmpb.2022.106749
0169-2607/

the state machine method [13]. Those two method split elements swept by the cutting tool into several smaller elements. Unfortunately, the element refinement method tends to generate narrow and flat tetrahedrons(called ill-shaped elements) and would easily lead to numerical instability. To avoid the numerical problem, Molino proposed a virtual node method [14], which creates one or more replicas of the elements that are cut and embeds each element belonging to a distinct material connectivity component into a corresponding replica. FEM using hexahedral elements [3] is another way to avoid ill-shaped elements. Hexahedral meshes can be directly generated from medical image data [15] or by voxelization techniques which use a unified hexahedral mesh to decompose the simulation object into a set of hexahedral elements. These methods can re-mesh the cut elements robustly and effectively. Recent research uses the extended finite element method (XFEM) in deformable models cutting simulation [16,17], which uses enrichment functions to ensure the deformation behaviour near the cut surface is physically correct. However constructing enrichment functions may be complicated in some complex cases.

Mesh-less technique are first proposed to reduce the dependence on the simulation grids. As the name implies, mesh-less methods do not require a simulation mesh or grid. The model is represented by a set of moving nodes, which interact with each other according to elasticity's governing equations. Cutting method for mesh-less objects typically inserts new nodes around the cut surface and updates visibility between nodes occluded by the cut surface [18,19]. This leads to three major shortcomings of the approach: adjacency information between nodes update at every step; additional data structure used to construct cut surface; numerical instability due to artificial discontinuities.

Commonly, the visual quality of analog objects is bound to the precision of geometrical representation, and the improvement of visual effect will be accompanied by the steady linear growth of the simulation unit. In order to solve these problems, many researchers have adopted the multi-resolution simulation framework. Nesme first introduces the concept of composite finite elements method (CFEM) [20]. CFEM embeds a high-resolution visual mesh into a fine hexahedral mesh. At the same time, a fine hexahedral mesh is also embedded into a coarse hexahedral mesh. CFEM performs branch analysis during cutting. So that coarse hexahedrons are assigned to the corresponding material component. Material components preserve the fine-level topology at the coarse level [21]. To enable high computational efficiency, Dick intertwines the cutting procedure and the deformation computation [22]. The algorithm uses the octree structure to perform adaptive element refinements and embed topological changes of the simulation grid into a geometric multigrid solver. By introducing the dual contouring method to update surface, Wu effectively eliminates jagged artifacts of the cutting surface in CFEM [23]. Besides, Wu uses the signed distance field to measure penetration depth to speed up the collision detection of CFEM [24]. To speed up collision processing further, Jia [25] splits the simulation frame into four independent tasks running in parallel based on Wu's method.

While most of the efforts have been devoted to solving the three major tasks of cutting simulation, cut fracture evolution modeling is still an open problem for deformable cutting simulation which greatly affects the realism of simulation results [1]. Most methods still follow a "Intersection-IS-Fracture" mode. To be specific, as long as an intersection is detected between the knife trajectory and the object, the simulation method will produce a new cut. This is quite different from the real cutting process of deformable objects. As most common soft objects have certain fracture resistance, an indentation aligning with the knife blade will appear before a fracture occurs. Even if the blade continues to press down on a cut already made, it will not always produce a cut. This phenomenon is called delayed fracturing [26]. A versatile cutting fracture evolution model is required to determine when and how a new cut occurs thus to achieve this real-life cutting effects. Although some primary models have been proposed [21], the geometry-based cutting fracture criteria is quite limited in both stability and scalability.

In this paper, we present an energy-based cutting fracture evolution model for cutting simulation of deformable objects. To facilitate medical applications, such as virtual surgery training and interactive surgical planning, we adopt the composite FEM (CFEM) [20] as the underlying deformation model due to its advantages in both efficiency and robustness. To focus on fracture modeling, the proposed method handles the collision and cut incorporation in the same way as the current FEM-based cutting methods using hexahedral elements [24,25]. A Griffith's energy based cutting fracture model is proposed to model the real-life cutting phenomenon. Even the Griffith's energy have been widely used for highly realistic ductile fracture simulation [27,28], it can't been directly applied to interactive virtual cutting simulation. The main reason is that most ductile fracture simulation methods focus on modeling the fracture phenomenon under twisting or stretching operations [28], without considering the interaction between cutting tool and object. Although some methods take this interaction into account using a threshold value to determine whether a cut fracture occurs [29,30]. But in reality, cutting fracture and ductile fracture are possible to occur simultaneously. These threshold-based methods are difficult to combine with energy-based ductile fracture models in further work, which limits the effect of more realistic fracture simulations. To solve this problem, we use CFEM and Griffith's energy to construct a fracture evolution model.

Because delayed fracturing is introduced, so the object no longer fractures immediately when a tool intersects with it. Besides, there should be a highly consistent indentation in the contact region. At the same time, the surface near the indentation will deform due to the contact force. However, this phenomenon is not well simulated in the interactive scenario, because of two limitations: simulation mesh resolution and simulation performance. First, the mesh resolution is always limited after discretization. In such a limitation, the blade often doesn't intersect the mesh vertices which results in the inconsistency between the generated indentation and the blade. So, we have to adjust the visual surface mesh to reduce the inconsistency. Second, classical physically-based approaches can deform the surface near the indentation realistically [31]. The obvious advantage of these approaches is their physical accuracy. However, they require a long computation time. This limits the usage of those pure physically-based approaches in an interactive context. Meanwhile, CFEM uses interpolation to control the position of the visual mesh, which leads to its inadequacy to produce sufficiently realistic indentation-induced deformation in real-time scenarios. Thus, we also need a time-independent scheme like the deformer in [32] to approximate the visual plausible deformation.

To reduce the negative impact of these two limitations, we propose a material-aware adaptation scheme to simulate highly consistent indentation and the visual plausible indentation-induced deformation. Together this scheme and the energy-based cutting fracture evolution model constitute a virtual cutting simulation algorithm with high realism for deformable objects.

The rest of the paper is organized as follows: the basic geometrical and physical model are introduced in Section 2, we introduce the cutting fracture evolution model designed for highly realistic soft object cutting in Section 3, the simulation results and performance evaluation are presented in Section 4. the paper is concluded in Section 5.

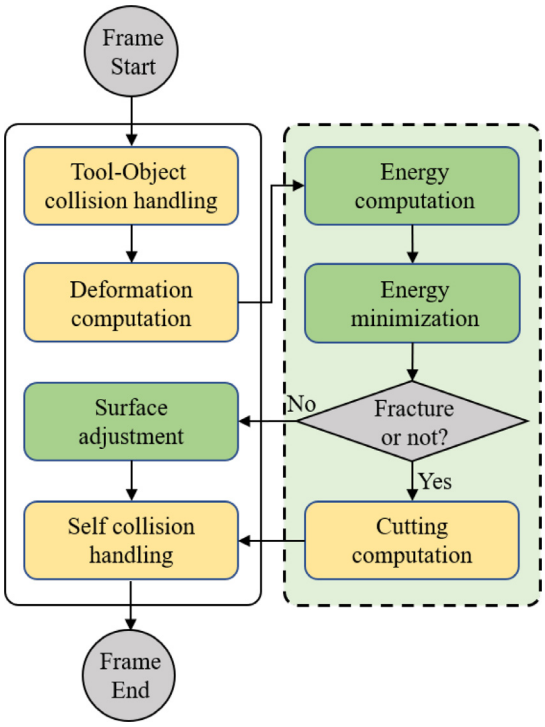


Fig. 1. Main steps in a single simulation frame. The three steps in green boxes are introduced by the proposed algorithm. The simulation loop starts with tool-object collision handling and deformation computation. Then it goes to the cut related computation, including energy computation and minimization and fracture determination based on the minimization result. If new fracture occurs, the algorithm conducts the conventional cutting computation, including geometric and topological updating. Otherwise, it will conduct surface adjustment to generate a plausible indentation. In the last step, self-collision will be handled.

2. Highly realistic virtual cutting simulation

To keep the potential for interactive medical applications, we design the virtual cutting algorithm based on CFEM because it can provide a good trade-off between simulation efficiency and accuracy and also can guarantee the numerical stability. Thus, the proposed method handles collision and cut incorporation in the same way as previous FEM-based methods using hexahedral elements [24,25]. To model real-life cutting effects, which can be considered as a "Press-Cut" repeating process, we introduce a Griffith's energy based cutting fracture evolution model to determine when and how the a new cut is generated. Besides, to guarantee the consistency between the cutting tool blade and the indentation which appears before a new cut occurs, the paper presents an iterative optimization scheme to efficient adjust the local surface mesh without introducing visible geometric. So here we will first introduce the basic cutting simulation framework. After that, basic model based on CFEM the cutting fracture evolution model and the material-aware adaptation scheme will be discussed in detail.

The main steps in each single simulation frame is as illustrated in Fig. 1. To incorporate the cutting fracture evolution model which requires to compute the Griffith's energy for the deformed object before a new cut occurs, each simulation frame starts from tool-object collision detection and deformation simulation. Meanwhile, surface mesh is adjusted to generate an indentation exactly aligned with the cutting tool blade. Then, the algorithm determines whether to produce a new cut. If fracture occurs, the geometrical and topological representations will be updated accordingly. In the last step, object self collision is detected and handled to prevent self-intersection artefacts. The remaining part of this section will introduce the main components of the basic cutting simulation framework.

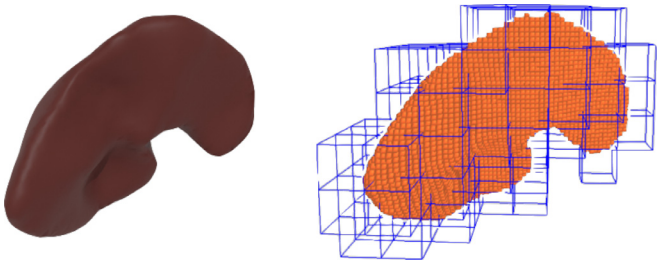


Fig. 2. Deformable simulation model. This picture shows an example of the geometric representation of a liver. On the left, there is the visual surface mesh. On the right, the orange hexahedrons are the fine voxel and the blue wireframes are composite elements.

2.1. Basic model based on CFEM

In this section we will introduce basic model based on CFEM. This part includes the geometric and topological representation, cut incorporation scheme and collision handling method.

Deformable simulation model: Three geometric components are employed to simulate cutting in deformable objects: fine voxel, composite elements and the surface mesh, see in Fig. 2, and a 2D illustration is also given in Fig. 3(a). Wherein, the composite elements (the blue wireframes in Fig. 2) govern the elastic property of the soft object and each 3D composite element is constructed by compositing $k \times k \times k$ blocks of fine voxel. The fine voxels(the orange hexahedrons in Fig. 2) approximate the given object by uniform Cartesian grid or octree grid. Links are assigned to each pair of face-adjacent fine voxels and are also used for topological representation. The original object surface are then embedded in the link grid by storing the intersection point and normal on each link intersected with the object surface. Dual contouring surface algorithm can be adopted to reconstruct the surface mesh(on the left of Fig. 2) for visualization [23]. In each simulation frame after the deformation computation on the composite elements, the displacements of the composite elements' vertices are further propagated to the fine voxels and then to the surface mesh vertices, both by means of trilinear interpolation. Deformable computation on the coarse composite elements can high reduce the number of simulation DOFs while the highly detailed surface mesh can be reconstructed from the link grid for high quality visual rendering.

Cut Incorporation: As discussed in the previous paragraph, the topology is represented by links between face-adjacent elements. Once a new cut occurs, the three geometric representations should be updated accordingly. Fig. 3(b) illustrates the updating process. First, to represent the topological discontinuity caused by the new cut, the correlated links should be marked as disconnected. If the cut separates the hexahedral elements in a composite element into several disconnected parts, duplicated individual composite elements should be created to model this disconnectivity. The connection among the affected composite elements should also be updated for correct physical simulation. Due to those complex topology changes, cutting simulation always needs much more computation to deal with topology modification. For generating the surface after the cut, the algorithm also need to re-calculate the intersection positions and normals on the links been cut.

Collision detection and handling: In cutting simulation, there exists two kind of intersections, namely the intersection between the cutting tool and object, and the self-intersection of topologically disconnected parts. We adopt the two-phases collision processing method proposed in [24]. A spatial hash table and a signed distance field corresponding to the deformable object are adopted

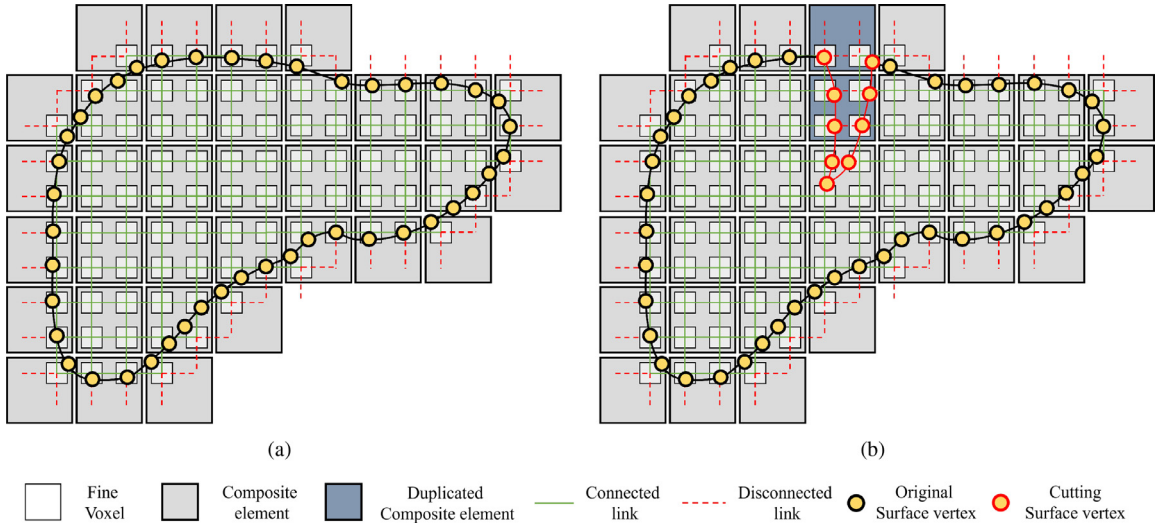


Fig. 3. Geometric representation for CFEM-based virtual cutting. (a) Geometric representation before cutting. Composite elements are used for deformation computation. Fine voxels and the links represents the topology of the object. Surface mesh is embedded on the link grid. (b) Geometric representation with cut incorporation. Links cut are marked as disconnected and composite elements divided into several independent parts are duplicated. New surface vertices are added by modifying the embedded data on the link grid.

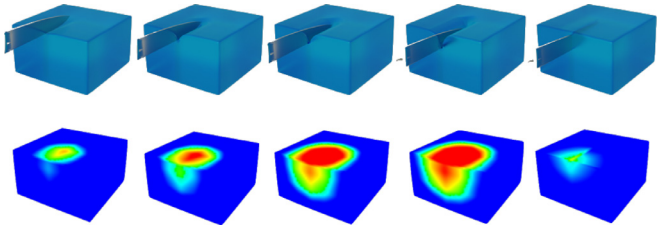


Fig. 4. Energy evolution visualization. This figure visualizes the energy evolution of cutting a cuboid. The first row shows the process from deformation to fracture, and the second row shows the corresponding energy visualization, which indicates the energy from low to high by the color from blue to red. As the tool is continuously pressed down, more and more energy is accumulated in the object until it fractures and energy will release.

for efficient broad phase collision detection and narrow phase collision detection respectively. Each hexahedral element is added to the hash table entries corresponding to the spatial cells its axis aligned bounding box (AABB) covers. For the tool-object collision, a line segment is used as the simplified cutting tool and in each time step, an AABB for the cutting tool trajectory is constructed. Together with the two-phases collision detection methods and the topology aware interpolation scheme [24], the algorithm can efficiently and accurately detect both tool-object collision and object self-collision. Afterwords, penalty forces proportional to the penetration depth are applied to the correlated composite elements. In addition to that, the signed distance field value has the ability to represent the surface. To fit the discretization modeling of the original CFEM we will use the signed distance field value of voxels to evolute links between them, in the Section 2.2.

2.2. Griffith's energy based cutting fracture evolution model

Evolution formulation: In the real-life cutting on a deformable object, the object would first deform under the force from the cutting tool. The work done by the tool is converted into the elastic energy of the object. Once a fracture occurs, part of the energy will convert to the irreversible work spent in creating the fracture and the remaining part is still in the form of elastic energy. As shown in Fig. 4, this process involves an evolution of energy and fracture. We use Griffith's energy theory to model this process. For simplicity, Griffith's energy at potential fracture surfaces can be defined as:

$$e_g = W_e + W_r, \tag{1}$$

where W_e and W_r are the elastic energy and the released energy cost in the fracture. W_r is related to the area created by the fracture, which can be calculated by the topological continuity and discontinuity of the object. In Sec2.1 we use voxel links to represent these two topological states of the object. Voxel links can represent whether a unit area fractures, see in Fig. 5. Once is a voxel link is disconnected, the released energy will increase while the elastic energy of the link will decrease. Thus, to adopt this equation to our framework, we approximate Griffith's energy of the potential fracture surface based on voxel links. That is, W_e and W_r are denoted by connected voxel links and disconnected voxel links, respectively. Using this link approximation, Eq. 1 can be rewritten as the integral at potential fracture surfaces:

$$e_g(\varphi) = \int_{\Omega_H} \Psi_l dX + \int_{\Omega_D} \kappa dX, \tag{2}$$

in which, Ω_H represents all the connected voxel links, and Ψ_l is the energy density for each voxel link. Ω_D represents all the disconnected voxel links, and κ is the energy release rate of the voxel links. The computation of the energy density Ψ_l will be described in the subsequent section. Since the simulation model is discretized, Eq. 1 can also be regarded as the following equation, which calculates Griffith's energy in our actual application:

$$e_g(\varphi) = \sum_{i \in \Omega_H} \Psi_l^i + \sum_{j \in \Omega_D} \kappa^j \tag{3}$$

We use a Griffith's energy minimization [33] as our evolution criterion. The fracture evolution process is simplified as evolving the connection state of voxel links. To apply minimization scheme while ensuring considering the tool-object interaction, we use the signed distance field value φ of the intersection point. It's located on each voxel link intersected with the cutting blade. φ of the intersection point can reflect whether the point is located on the inner or outer side of the object surface. For example, $\varphi > 0$ indicating that the intersection point is on the outside of the object. In cutting process φ will change. For each voxel link intersected with the cutting blade, our strategy is to evolve φ to estimated φ_{new}. If

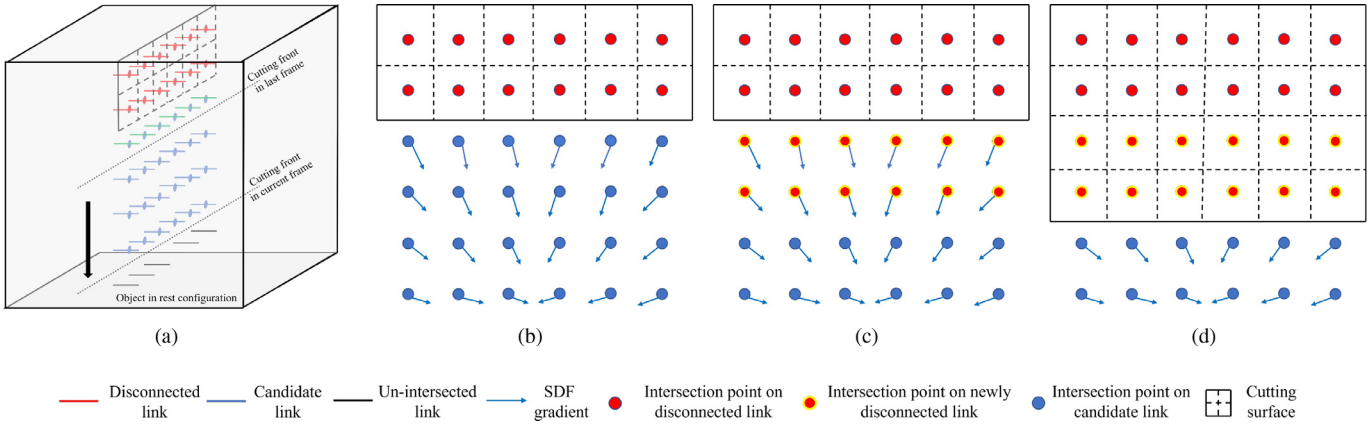


Fig. 5. Cutting plane constrained Griffith's energy minimization. (a) Collect the links intersected with the cutting plane in the current frame. (b) Compute the Frechet derivatives for each intersection point on the links. (c) New links are disconnected after one step of the gradient descent approach. (d) The minimization results and the updated cutting front which is aligned with the cutting blade.

$\varphi_{new} > 0$ the voxel link should be disconnected. Thus,we can determine the connection state.

We need to construct a forward-Euler scheme to evolve φ of the intersection point

$$\varphi_{new} = \varphi + \Delta s \frac{\partial \varphi}{\partial s} \tag{4}$$

in which, s is the pseudo parameter of time evolution.

We use the gradient descent approach to solve this problem as previous ductile fracture simulation algorithm. As derived by [27], Frechet derivative of Eq. 2 in the direction of $\delta\eta$ is

$$\int_{\partial\Omega_H} (\kappa - \Psi)\delta\eta dA, \tag{5}$$

in which, $\delta\eta$ is chosen as the direction in which the signed distance field value drops fastest. Then, using Eq. 5 in a gradient descent approach

$$\frac{\partial \varphi}{\partial s} = -\delta(\varphi)(\kappa - \Psi), \tag{6}$$

Substituting the Eq. 6 into Eq. 4, we obtain a forward-Euler scheme to update the signed distance field value of voxels whose link is intersected with the cutting plane. The delta function $\delta(\varphi)$ restrict the evolution to the surface.

Computation of the energy: The elastic energy of a deformed object can be modeled by the integral of the energy density on each finite element. In the CFEM-based cutting framework, the physical computation is conducted on the composite elements and the topology information is represented by the links among fine voxels. Accordingly, the energy density can be first calculated on the composite element and then interpolated onto the fine voxel links. Given an object with region Ω_0 in the initial undeformed configuration, the energy density in each position in the object at time t with the deformed region Ω_t can be defined based on the deformation gradient:

$$F(X,t) = \frac{\partial \Phi}{\partial X}(X,t), \tag{7}$$

where $\Phi(X,t)$ is the mapping function between positions in the initial state and the deformed state at time t, and X is an arbitrary position in Ω_0.

For simplicity we will here consider the energy density Ψ as a function of the deformation gradient. Specifically for co-rotational elasticity, we obtain the energy density function

$$\Psi(\mathbf{F}) = \mu\|\boldsymbol{S} - \mathbf{I}\|_F^2 + (\lambda/2)\,\mathrm{tr}^2(\boldsymbol{S} - \mathbf{I}) \tag{8}$$

S is the symmetric tensor from the polar decomposition $F = RS$. In 3D simulation, the coefficients μ and λ correlated to the FEM coefficients young's module E and poisson ration ν by the following equation:

$$\mu = \frac{E}{1+\nu}, \lambda = \frac{E}{1-2\nu} \tag{9}$$

After calculating the energy density on the nodes of the composite element, we can get the energy density on each link by interpolating. The sum of energy on all connected links can be regarded as the elastic energy of the deformed object.

On the other hand, according to Griffith's energy theory, W_r in Eq. 1 is determined by two factors: the area of the new generated cutting face and the energy release rate. By assuming that associated cutting area for each link is equal to a unit area, the total energy cost for generating a new cut is the sum of all the κ correlating to the cut links. Thus, we define the energy release rate coefficient κ for each voxel link. Varying κ can be assigned to an object to produce cutting effects of heterogeneous material.

2.3. Material-aware adaptation scheme

The material-aware adaptation scheme consists of two main steps, the effect as showed in the third row of Fig. 9. First, the low resolution of the simulation mesh leads to visual inconsistency artifact between the cutting tool blade and indentation. To solve this problem, the surface vertices that may stay in contact with the blade are defined as the contact region. Some vertices in the contact region need to be projected onto the cutting blade. This step produces an aligned indentation. To reduce the distortion, we define the algebraic quality measure for each triangle mesh. By minimizing the algebraic quality measure, vertices are projected onto the optimal position. Meanwhile, simulation performance causes computation time to be too short to produce the visually plausible indentation-induced deformation. To reduce the discontinuity caused by this problem, we deform the neighbor surface of the indentation plausibly. To achieve this goal, a set of neighbor surface vertices are added into the deformation region by the geodesic distance. Then in this region, we use a parametric B-spline to instantiate the deformation profiles.

Indentation generation: We first generate the indentation by contact region. It's determined by the intersections of the blade and the object. Some vertices of the contact region will stay in contact with the blade. These vertices that called indentation vertices form the indentation. To generate a consistent indentation, we approximate the blade curve C with a series of control points. We aim to find a suitable mapping to move some vertices of this region to control points, which is called projection. The mapping

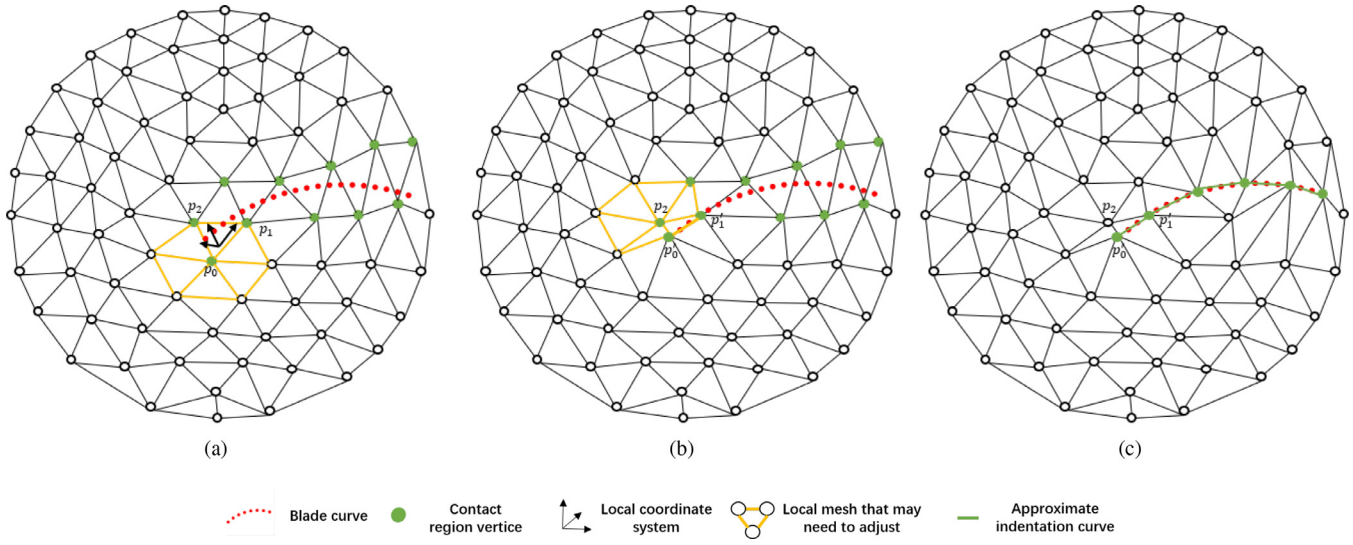


Fig. 6. Approximate indentation curve. (a) For each contact region vertice, the local mesh that containing the vertice is identified. A local coordinate system is established to measure the algebraic quality of those triangular meshes. (b) For the contact region vertices, iterations are performed on the blade curve to find an optimal projection point that minimizes the quality function and satisfies the constraint. (c) After projection, perform the process for the rest contact region vertices until the adjustment is finished.

can be found by some simple constraints, such as the shortest distance. In this process, vertices project to a control point, which causes the shape of surface triangular meshes to change. The simple constraints in the conventional method may cause inverted or ill-shaped triangular meshes. To avoid this situation, we should find an optimal projection position for vertices by constructing a suitable constraint. Thus we need to evaluate the quality of the changed triangular meshes.

We first construct jacobian matrixes for the changed triangular meshes for a given vertice by parameterizing them in two dimensions. Then, using the matrix properties, construct an algebraic quality measure Q for each triangular mesh that may be changed. The next step is to find a projection position for the given vertice to ensure that the sum of Q is minimum. This position is the optimal projection position for the given vertice. The scheme consists of main steps as illustrated in Fig. 6.

The first step requires a local coordinate system of the triangular mesh to simplify the problem. In three-dimensional space, giving a triangular mesh t, whose three vertices are p_0, p_1 and p_2. Furthermore, the coordinate axes of the local coordinate system can be expressed as:

$$\boldsymbol{X} = \frac{p_1 - p_0}{|p_1 - p_0|} \quad \boldsymbol{Z} = \frac{X \times (p_2 - p_0)}{|X \times (p_2 - p_0)|} \quad \boldsymbol{Y} = Z \times X \tag{10}$$

With this local coordinate system, the local two-dimensional coordinate of the vertex p_i can be expressed as $p_i' = (x_i, y_i)^T, i = 0, 1, 2$ Let t_R with vertices $r_0 = (0,0)^T, r_1 = (1,0)^T, r_2 = (0,1)^T$ be the reference triangle. Choosing p_0 as the translation vector, the affine map that translates t_R to t is $p' = p_0 + Ar$ where A is the Jacobian matrix of the affine translation map $t_R \overset{A}{\rightarrow} t$:

$$A = \left(p_1' - p_0', p_2' - p_0'\right) = \begin{bmatrix} x_1 & x_2 \\ y_1 & y_2 \end{bmatrix} \tag{11}$$

To measure the quality of t an ideal triangle is needed. In our case, it is the equilateral triangle. Let t_I with vertices $\mathrm{w}_0 = (0,0)^T, \mathrm{w}_1 = (1,0)^T, \mathrm{w}_2 = (1/2, \sqrt{3}/2)^T$ be the ideal triangle. Choosing w_0 as the translation vector, the Jacobian matrix of the affine translation map $t_R \overset{W_I}{\rightarrow} t_I$ is defined as W_I:

$$W_I = (w_1 - w_0, w_2 - w_0) = \begin{bmatrix} 1 & 1/2 \\ 0 & \sqrt{3}/2 \end{bmatrix} \tag{12}$$

Then, we define H as the weighted Jacobian matrix of the affine translation map $t_I \overset{H}{\rightarrow} t$:

$$H = AW_I^{-1} \tag{13}$$

Employing the Jacobian matrix H, which reflects the change of t relative to t_I, we can construct an algebraic quality measure of triangular mesh t. Let $P = (x, y)^T$ be the position vector of the vertice P, which is needed to be adjusted. N triangles containing vertice P are expressed as $t_n, n = 0 \ldots N-1$ and H_n is the weighted Jacobian matrix of t_n. These triangles constitute the local mesh $N(P)$ (the yellow mesh in Fig. 6(a)). The quality function for the local mesh is

$$Q(P) = \sum_{n=1}^{N} \left(\frac{F_n}{2D_n}\right)(P) \tag{14}$$

Where D_n is the determinant of H_n, reflecting the translation of t to the ideal triangle. Meanwhile F_n is the Frobenius norm of H_n.

Using this quality function, we can find an optimal projection point for P on the blade curve C. The method is to move P and project it onto the blade control points on curve C, and find the optimal projection position by minimizing this formula:

$$\text{minimize}\left[Q\left(P_{\text{project}}\right)\right], \text{ subject to } P_{\text{project}} \in C \tag{15}$$

It should be noted that t_n is inverted if $D_n \leq 0$. To prevent this situation, we conclude $P_{project}$ is the optimal projection position only if $P_{project}$ is the position minimizing the formula when all $D_n > 0$. If so, vertice P will move to $P_{project}$, as P_0 in Fig. 6(b). Otherwise, vertice P is not currently projectable.

Indentation-induced deformation generation: After generating indentation, this step aims to produce the plausible indentation-induced deformation in elastic surface around the indentation. Otherwise, a discontinuity may appear, see in the second row of Fig. 9. The indentation-induced deformation is related to the object's material. The softer the material, the more pronounced the deformation will be. To produce the deformation related to the material, we develop a material-aware scheme to approximate deformation. We find surface mesh vertices that belong to the deformation region and construct a shape function to compute the approximate position of these vertices, see Fig. 7.

At first, we use the mesh's topology and the geodesic distance to find the adjacent vertices of the indentation vertices, as the

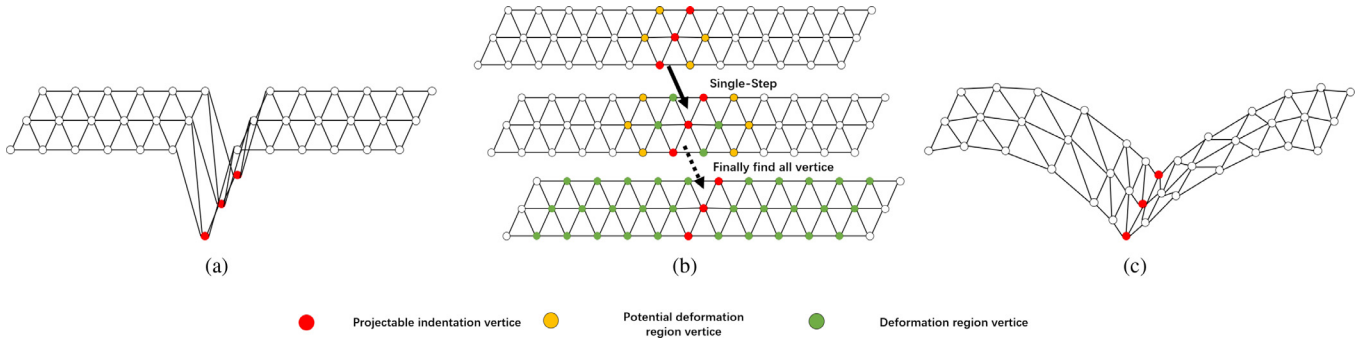


Fig. 7. Indentation-induced deformation generation. (a) After indentation generation, a serious discontinuity would appear. Therefore, the deformation region needs to be determined for further adjustment. (b) We use the mesh's topology to find the potential deformation region of the indentation region by parameterizing. (c) Then we use parametric B-spline to produce the deformation profiles in the deformation region.

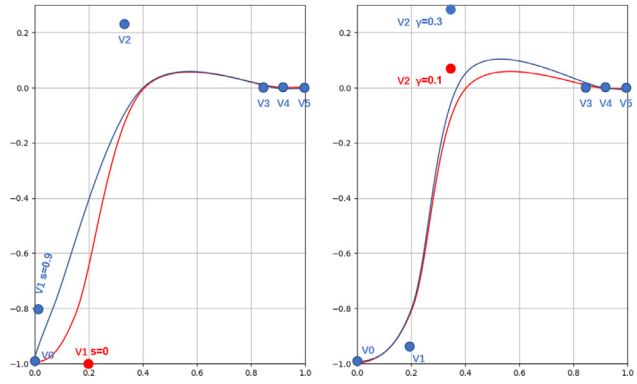


Fig. 8. Material-aware parameters. A simple two-dimensional cross-sectional plot showing the effect of the parameters on the curve. On the left, The position of v_1 is constrained by s, while the position of v_2 is according to γ on the right. For surface continuity, v_0 is fixed.

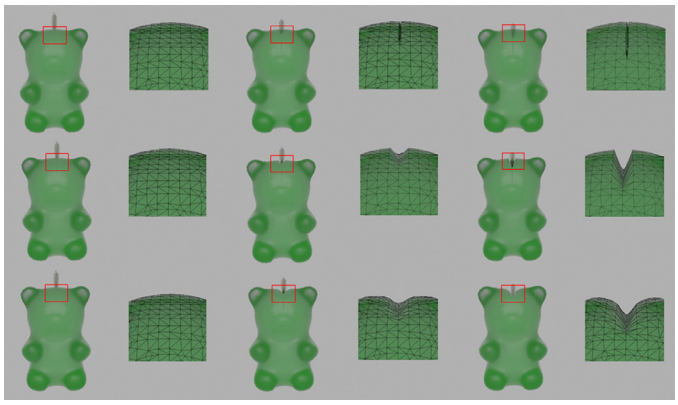


Fig. 9. Comparison with the basic CFEM cutting. The first row shows the effect of basic CFEM cutting, and the second row shows the intermediate result of fracture evolution and indentation generation. The third row shows the complete effect of our algorithm. The image in each row consists of the front view of the cut effect and the corresponding zoomed-in view.

Fig. 7(b) llustrates. The deformation region is bounded by geodesic distance from the indentation. For each indentation vertice, we find its adjacent vertices, then parametrize their adjacent level l to the indentation vertice. The vertices with $l < 1$ constitute the deformation region, and l is calculated by the following equation:

$$l = \frac{g}{g_{\max}} \tag{16}$$

With g is the geodesic distance between indentation vertice and corresponding adjacent vertice. $g_{\max}$ is max geodesic distance controlling the extent of the deformation. The deformation area is related to the material. The softer the material, the larger the deformation area. Thus, $g_{\max}$ is determined by the material:

$$g_{\max} = \frac{V}{E} \tag{17}$$

V is a simulation scale-related empirical parameter.

The next step is to produce the deformation profiles in the deformation region, see Fig. 7(c). We use open uniform B-spline for their smoothness and local control. The B-spline $f_{B-spline}$ needs to adjust each vertex p_i of the deformation region to match the indentation and material. To match the indentation, the projection of corresponding indentation vertice is introduced as the amplitude a_i. Meanwhile, we use slope s_i and bulging parameter γ to match simulation for different materials, see in Fig. 8. s_i reflects the steepness of the deformation, which is related to the material. It can be approximated as the ratio of $g_{\max}$ to a_i. While γ produce pseudo-volume preservation. It is determined by empirical parameter τ and young's module E:

$$s_i = \frac{a_i E}{V}, \gamma = \frac{\tau}{E} \tag{18}$$

In our experiments the value of τ is set as 2×10^5. Using these parameters, we can define $f_{B-spline}$ by six control points $v_n, n \in \{0 \ldots 5\}$ in below:

	v_0	v_1	v_2	v_3	v_4	v_5
abscissa	0	$0.2\cos\theta_i$	1/3	5/6	11/12	1
ordinate	$-a_i$	$a_i(0.2\sin\theta_i - 1)$	γ	0	0	0

With $\theta_i = \tan^{-1}(s_i/a_i)$ and v_3, v_4, v_5 ensure elastic surface continuity at $l \approx 1$. Using $f_{B-spline}$, we can smooth the surface by moving p_i:

$$p_i' = f_{\mathrm{B-\ spline}}(l_i)d_i + p_i \tag{19}$$

l_i is the adjacent level of p_i, and d_i is unit vector of the corresponding indentation vertice's displacement vector.

3. Experimental results

In the following, to prove the versatility of the proposed algorithm, we conducted several experiments on different models, materials, and tools with different shapes. A video showing these qualitative evaluation is provided in supplemental material. At last we give the performance statistics and an execution step timing breakdown.

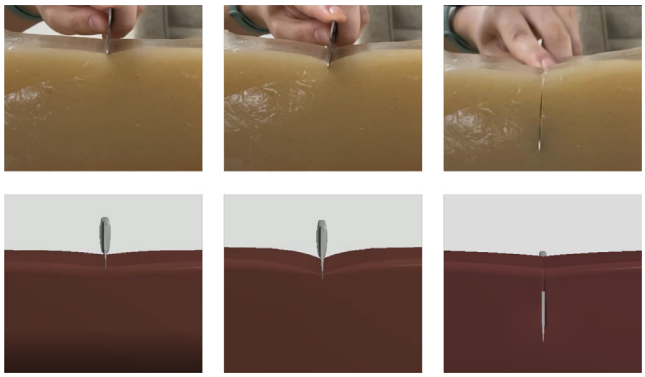


Fig. 10. Comparison to realistic cutting experience. The left side shows the cutting process in the real scene, and the right side shows the simulation results of our method.

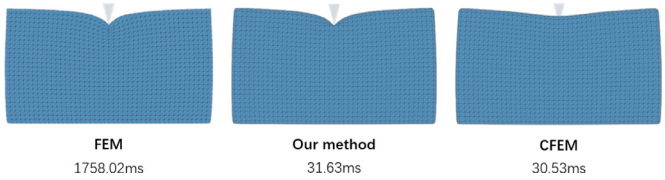


Fig. 11. Comparison of simulation results. From left to right are the pressing simulation results of the FEM, our method, and CFEM, respectively. Below each result is its corresponding average execution time.

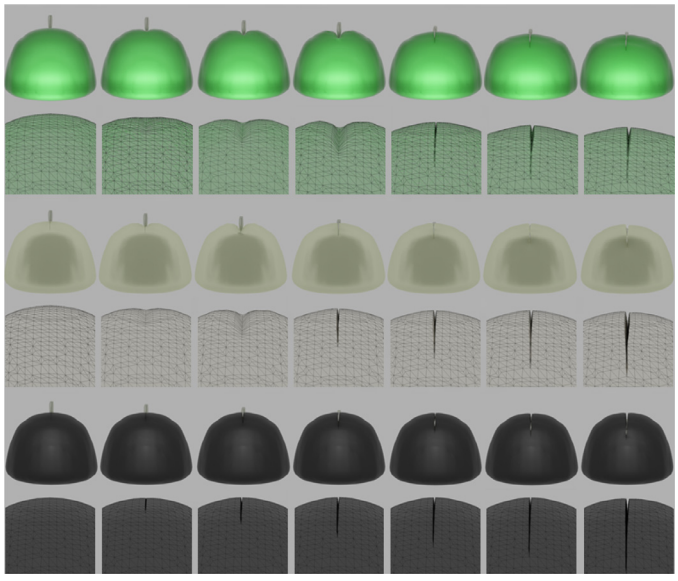


Fig. 12. Cutting effect of different materials. This picture shows simulation effects for three different materials. Every two rows with a similar color represent the simulation effect of one material. The bottom row of these two rows shows the corresponding zoomed-in view.

The proposed algorithm is implemented in C++ with Visual studio 2013 and runs on a PC with the central processing unit i7-6700HQ, 16 GB RAM, and a GTX 960M graphics card. A Phantom Omini is also used in the experiments for applying arbitrary cutting.

Comparison of cutting effect: To reflect the visually realistic of the proposed method, we compare its simulation result with the actual cutting experience in Fig. 10 and compare with CFEM which is the baseline in Fig. 9. To prove the enhancement of the simulation effect, we compare our algorithm with the basic CFEM cutting using the same model in Fig. 9.

In the real scene, as the cutting blade intersects with the deformable object, a new fracture doesn't occur immediately, while the deformable object firstly deforms and appears an indentation aligning with the blade before a fracture occurs. As the blade goes deeper and deeper, the object eventually fractures. The first row in Fig. 9 shows the effect of basic CFEM cutting in chronological order from left to right. The cutting fracture occurs as long as the cutting blade intersects with the model, which is different from the realistic cutting experience. The second row in Fig. 9 shows the intermediate result of fracture evolution and indentation generation. Despite the introduction of delayed fracture, the results are still not realistic enough. Because there is a significant discontinuity in the elastic surface around the indentation. It is caused by the lack of indentation-induced deformation. Using pure physically-based deformation approaches to generate indentation-induced deformation in an interactive context is limited because it would greatly reduce the simulation efficiency. Thus, a geometric approximation scheme is used to produce indentation-induced deformation in the proposed algorithm. By combining fracture evolution and material-aware adaptation scheme, we can obtain the complete effect of our algorithm in the third row Fig. 9, which is more similar to the realistic cutting experience.

In the Fig. 10, we cut a deformable object in the real scene and we simulate this cutting process using the proposed method. As in the real scene, the simulation result first deform and appear an indentation aligning with the knife blade before a fracture occurs. The simulation result of the proposed method presents a similar "Press-Cut" cutting process as the real scene, which reflects the visually realistic of our method. More details are given in the supplemental video.

Comparison of indentation effect: To demonstrate the effectiveness of the surface adaptation, in this experiment we use different methods to simulate the process of a given object being pressed. The visual surface meshes all have the same resolution $43 \times 43 \times 27$. In this case, FEM is calculated directly on the visual surface meshes to determine their position. In contrast, our method and CFEM controls the position of the visual surface meshes by using the interpolation of composite elements. In the Fig. 11, the FEM provides result with high physical accuracy, but its average execution time reaches 1758ms, which is unacceptable in an interactive context. Although the execution time of CFEM is 30.53ms, its generated result are not accurate enough compared to the FEM. The result of our method is similar to the result of FEM, meanwhile, the execution time is not significantly increased compared with CFEM, so our method makes trade-off between simulation efficiency and visual effect.

Simulation results of different materials: In real life, materials are varied, and the cutting effect of the model changes with its material. The fracture evolution of our method is determined by the material parameters, including E and κ. It is possible to achieve cutting simulation effects for different materials. The effects of varying parameters on the same model are shown in Fig. 12. For visual effect, these models are assignment with colors similar to their materials in real life. In Fig. 12, the first, third, and fifth rows show the simulation effect for different materials and the second, fourth, and sixth rows show the zoomed-in view of the corresponding boxed area. In Fig. 12, the first row shows the simulation result of jelly, a material that would undergo large deformation under force and usually fractures after large deformation. The third row is the simulation result of tofu material, which is more resistant to deformation than jelly and fractures under less deformation. The fifth row shows the simulation result of rubber, which produces a little deformation when the cutting fracture occurs.

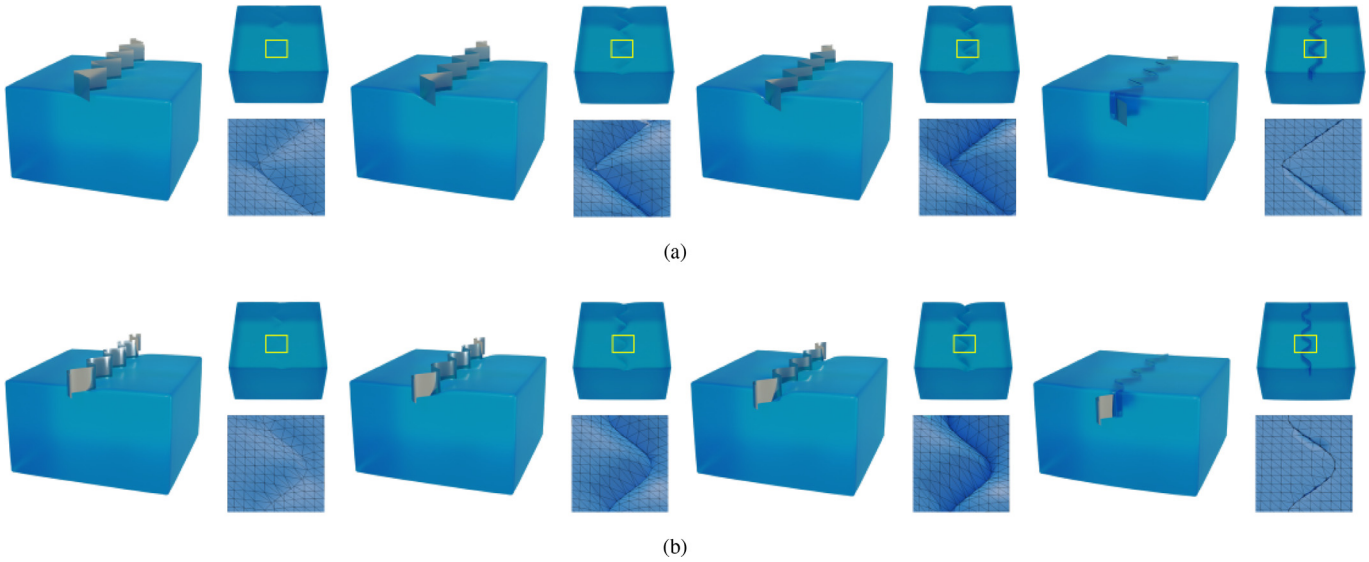


Fig. 13. Simulation results of different tools. These two pictures show simulation results of tools with different shapes. Both of them include the cutting process from deformation(the first to the third) to fracture(the last). On the left side of each picture in (a) and (b) is a side view of the cut effect, and on the right side is its corresponding top view with a local zoomed-in view that shows the details better.

The comparison between different materials illustrates the versatility of the proposed algorithm to the materials parameters. More details are given in the supplemental video.

Simulation results of different cutting tools: Our method can produce indentation which is highly consistent with the cutting tool blade. In this experiment, we used two types of tools with different geometric features to cut the model. The shape of the tool is a periodicity fold line or a trigonometric curve.

In Fig. 13(a) and Fig. 13(b), the simulation results are shown from left to right including the cutting process from deformation(the first to the third) to fracture(the last). When deformation occurs, the simulation results show whether the geometric feature of the tool is sharp (Fig. 13(a)) or smooth (Fig. 13(b)), our algorithm fits the indentation well and does not introduce the inverted mesh. As the tool gradually penetrates deeper and deeper, the model eventually fractures. The fracture also matches the blade no matter what shape the blade is. The fracture evolution process of our method ensures this consistency well.

Our method has good versatility to tools with different sharpness and smoothness. More details are given in the supplemental video.

Simulation result of hepatectomy: To demonstrate the potential application of the method geared toward actual surgery, we also performed a simulation of hepatectomy in Fig. 14. The black diseased tissue is first fixed by the tool and then it deforms firstly upon knife contact. As the knife gradually penetrates deeper and deeper, it eventually fractures. The whole process reproduces the realistic cutting and the average execution time is within 25ms, which can meet the real-time requirement well. More details are given in the supplemental video.

Simulation results of arbitrary cutting: To further validate the versatility and the robustness of our algorithm, arbitrary cutting operations are performed on models with different geometric features. The arbitrary cutting experiments are set up for different models including horse (Fig. 15(a)), armadillo (Fig. 15(b)) and bunny (Fig. 15(c)). We manipulate the cutting tool by a Phantom Omini to perform multiple arbitrary cutting operations on the above model. Screen captures of these simulation experiments are shown in Fig. 15.

The experimental results show the versatility to different models. Our method shows good robustness when cutting arbitrary parts of different models with different geometric features. More details are given in the supplemental video.

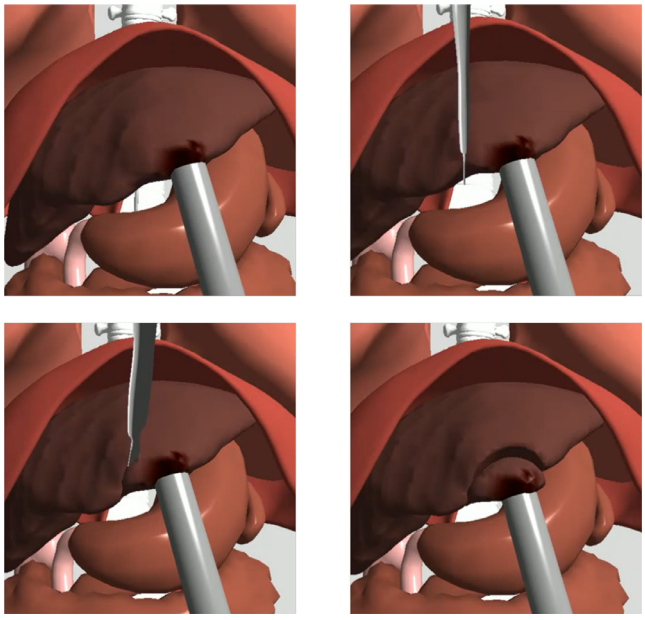


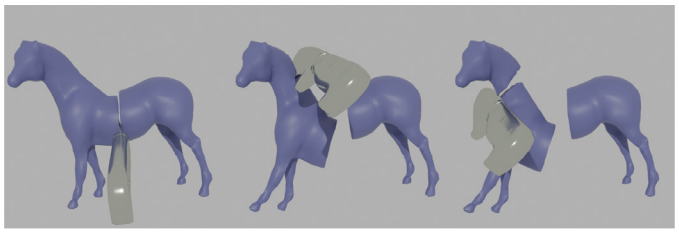
Fig. 14. Simulation result of hepatectomy. This image shows the visual effect of this method to simulate hepatectomy, where the black part is the diseased tissue.

Performance Analyses: Our method can be decomposed into several subroutines, in which fracture evolution and surface adaptation are proposed by our work. The rest of them belongs to basic CFEM cutting. In order to verify the two added subroutines whether introduce enormous computational costs, we record runtime and the actual number of executions for each subroutine to calculate their average runtime. The results are given in Table 1, and detailed resolution and material parameters of the models used in our experiment are also shown in this table.

According to the fracture evolution, our algorithm has two states: no fracture and fracture. Both of them execute FEM calculation, collision detection, and fracture evolution. No fracture state executes surface adaptation while fracture state executes topology

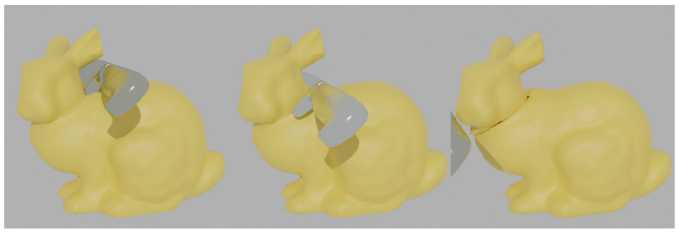
Table 1
Performance statistics.

Model	Voxels Resolution	Fine Voxels	Composite Elements	Surface Mesh	Material Parameters		Simulation Updating (ms)			Cut Updating (ms)		Totoal (ms)	
					Young's Modulus	Energy Release Rate	FEM	Evolution	Collision	Topology Modification	Surface Adaptation	Fracture	No Fracture
Gummy Bear	39 × 59×37	51,034	899	20,602	2.0MPa	0.55	20.42	1.89	7.48	30.17	2.83	59.96	32.62
Jelly	30 × 22×30	12,676	240	6656	1.0MPa	0.25	4.66	0.50	2.63	15.76	0.73	23.55	8.52
Tofu	30 × 22×30	12,676	240	6656	2.5MPa	0.40	4.68	0.49	2.70	14.46	0.64	22.33	8.51
Rubber	30 × 22×30	12,676	240	6656	8.0MPa	5.00	4.73	0.45	2.64	14.41	0.70	22.23	8.52
Cuboid(polyline)	46 × 28×46	59,248	144	18,128	1.5MPa	1.10	2.91	1.93	5.81	40.60	6.73	51.25	17.38
Cuboid(sin)	46 × 28×46	59,248	144	18,128	1.5MPa	1.10	3.08	2.04	5.95	41.70	7.26	52.77	18.33
Liver	58 × 48×40	30,360	559	16,804	2.0MPa	0.55	12.21	1.79	6.18	20.22	1.17	40.41	21.36
Horse	28 × 23×14	4828	68	4266	2.0MPa	0.55	2.17	0.32	3.09	4.24	0.32	10.14	5.90
Armadillo	46 × 54×40	16,965	109	13,520	2.0MPa	0.55	3.11	0.58	5.73	16.94	1.67	26.36	11.09
Bunny	56 × 56×44	44,221	158	21,576	2.0MPa	0.55	4.30	1.44	8.47	31.77	2.89	45.98	17.10



(a) Horse

(b) Armadillo



(c) Bunny

Fig. 15. Simulation results of different models for arbitrary cutting operations.

modification. Thus runtime varies in different states, the average runtime of them is also shown in Table.. The fracture evolution introduces a negligible cost compute time in both states. In no fracture state, surface adaptation introduced in our method has a comparatively negligible impact on the overall performance. In fracture state, the total runtime is larger, because topology modification is inherently the more time-consuming part in basic CFEM cutting.

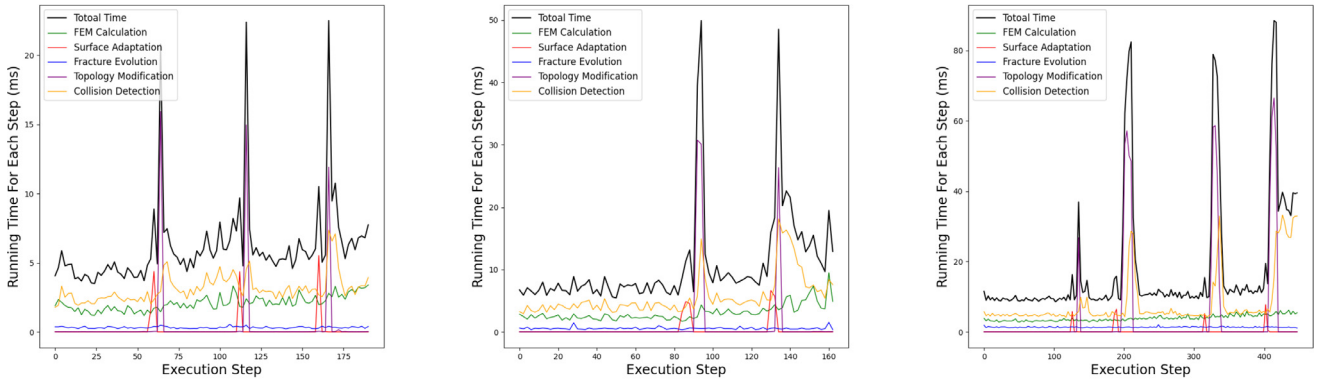
Note that the efficiency of topology modification and surface adaptation is also inevitably related to the actual contact area of the blade. As in Fig. 13, the blades have a larger contact area with the upper surface of the cuboids. Therefore algorithm needs to perform calculations on a larger number of fine voxels and surface meshes in experiments of Fig. 13. Thus their average runtime is longer than other experiments' runtime with the approximate resolution of fine voxels in the Table 1.

Performance of different resolution: To evaluate the limitation of the model resolution to achieve an interactive frame rate, we simulate the cutting process of a liver model with different voxel resolutions. The performance is shown in the Table 2. For easier evaluation, we use a uniform composite elements resolution, i.e., 64 voxels in each composite element. It can be seen that the efficiency gradually decreases as the resolution of the voxels increases. At this resolution comes to 30360, the average execution time is 21.64 ms, and the max execution time is also less than 41.67 ms. It ensures an interactive frame rate for the whole simulation process. Therefore, this resolution can ensure real-time and realistic performance for the liver model. Note that this suitable resolution varies for different models cause of their different shapes.

Timing breakdown: In Fig. 16, we present a timing breakdown for every single substep in experiments of Fig. 15. Different colored lines represent different subroutines. Fracture evolution(blue line) remains stable during the execution and has a very small effect

Table 2
Performance of different resolution.

Voxels	Composite Elements	Max Execution Time (ms)	Min Execution Time (ms)	Average Execution Time (ms)
991	29	2.62	1.61	1.63
5001	107	10.58	5.14	5.67
30360	559	41.24	20.43	21.64
55706	998	72.15	35.28	36.99



(a) Horse (b) Armadillo (c) Bunny

Fig. 16. We count the elapsed time of each subroutine and generate a timing breakdown. Different colored lines represent different subroutines. The result shows our fracture evolution (blue line) and surface adaptation (red line) add the comparatively low compute time to basic CFEM cutting. Timing are performed on (a)Horse (b)Armadillo (c)Bunny in Fig. 15.

on the total time(black line). Surface adaptation(red line) mainly executes before the fracture occurs and adds no compute time at the rest of the time, which is consistent with our algorithm. As the essential subroutines of basic CFEM cutting, topology modification(purple line) and collision detection(yellow line) have a decisive impact on the total time, while surface adaptation introduces comparatively low compute time. The compute time added by fracture evolution and surface adaptation is negligible, which highlights that our algorithm provides a good trade-off between simulation efficiency and accuracy.

4. Summary

In this paper, we present a highly realistic virtual cutting simulation algorithm by introducing an energy-based cutting fracture evolution model. To coincide with real-life cutting effects, we introduce Griffith's energy into basic CFEM cutting to determine when and how the new cut is generated. Also, to eliminate the visual inconsistency artifact due to limited resolutions, we propose a material-aware adaptation scheme.

Our algorithm can produce realistic cutting simulation results of different models, materials, and tools with different shapes, which indicates the versatility of the proposed algorithm. The experimental results show that our algorithm provides realistic visual effects of the "Press-Cut" repeating process in real-life cutting with a small-time cost, which keeps the potential of our algorithm for interactive applications.

Although our algorithm explores energy-based cut fracture evolution modeling on a single object and has a realistic visual effect. However, in practice medical applications, the scenario is very complex. It may contain multiple organs and has higher requirements for efficiency. We would further improve the efficiency by adopting parallelization techniques.

Declaration of Competing Interest

No conflict of interest has been declared by the authors.

Acknowledgements

This work was supported by Key-Area Research and Development Program of Guangdong Province, China (2020B010165004), National Natural Science Foundation of China (62072452, 61902386), Shenzhen Science and Technology Program (No. JCYJ20200109115627045, No. JCYJ20200109114233670 and No. JCYJ20180507182410327), Natural Science Foundation of Guangdong Province (2020A1515010357, 2021A1515011869), National Natural Science Foundation of China of Civil Aviation Joint Research Fund (U2133208).

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.cmpb.2022.106749.

References

[1] J. Wu, R. Westermann, C. Dick, A survey of physically based simulation of cuts in deformable bodies, in: Computer Graphics Forum, volume 34, Wiley Online Library, 2015, pp. 161–187.
[2] J. Zhang, Y. Zhong, C. Gu, Deformable models for surgical simulation: a survey, IEEE Rev Biomed Eng 11 (2017) 143–164.
[3] S.F. Frisken-Gibson, Using linked volumes to model object collisions, deformation, cutting, carving, and joining, IEEE Trans Vis Comput Graph 5 (4) (1999) 333–348.
[4] M. Bro-Nielsen, S. Cotin, Real-time volumetric deformable models for surgery simulation using finite elements and condensation, in: Computer graphics forum, volume 15, Wiley Online Library, 1996, pp. 57–66.
[5] X. Wu, M.S. Downes, T. Goktekin, F. Tendick, Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes, in: Computer Graphics Forum, volume 20, Wiley Online Library, 2001, pp. 349–358.
[6] S. Capell, S. Green, B. Curless, T. Duchamp, Z. Popović, A multiresolution framework for dynamic deformations, in: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation, 2002, pp. 41–47.
[7] M. Müller, B. Heidelberger, M. Teschner, M. Gross, Meshless deformations based on shape matching, ACM transactions on graphics (TOG) 24 (3) (2005) 471–478.
[8] E. Sifakis, T. Shinar, G. Irving, R. Fedkiw, Hybrid simulation of deformable solids, in: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation, 2007, pp. 81–90.
[9] M. Bro-Nielsen, Finite element modeling in surgery simulation, Proc. IEEE 86 (3) (1998) 490–503.

[10] H.-W. Nienhuys, A.F. van der Stappen, Combining finite element deformation with cutting for surgery simulations, in: EuroGraphics Short Presentations, 2000, pp. 43–52.
[11] D. Bielser, V.A. Maiwald, M.H. Gross, Interactive cuts through 3-dimensional soft tissue, in: Computer Graphics Forum, volume 18, Wiley Online Library, 1999, pp. 31–38.
[12] D. Bielser, M.H. Gross, Interactive simulation of surgical cuts, in: Proceedings the Eighth Pacific Conference on Computer Graphics and Applications, IEEE, 2000, pp. 116–442.
[13] D. Bielser, P. Glardon, M. Teschner, M. Gross, A state machine for real-time cutting of tetrahedral meshes, in: 11th Pacific Conference onComputer Graphics and Applications, 2003. Proceedings., IEEE, 2003, pp. 377–386.
[14] N. Molino, Z. Bao, R. Fedkiw, A virtual node algorithm for changing mesh topology during simulation, ACM Transactions on Graphics (TOG) 23 (3) (2004) 385–392.
[15] Y. Zhang, C. Bajaj, B.-S. Sohn, 3D finite element meshing from imaging data, Comput Methods Appl Mech Eng 194 (48–49) (2005) 5083–5106.
[16] L.M. Vigneron, S.K. Warfield, P.A. Robe, J.G. Verly, 3D XFEM-based modeling of retraction for preoperative image update, Computer Aided Surgery 16 (3) (2011) 121–134.
[17] C. Paulus, S. Suwelack, N. Schoch, S. Speidel, R. Dillmann, V. Heuveline, Simulation of complex cuts in soft tissue with the extended finite element method (x-FEM), Preprint Series of the Engineering Mathematics and Computing Lab (02) (2014).
[18] D. Steinemann, M.A. Otaduy, M. Gross, Splitting meshless deforming objects with explicit surface tracking, Graph Models 71 (6) (2009) 209–220.
[19] N. Pietroni, F. Ganovelli, P. Cignoni, R. Scopigno, Splitting cubes: a fast and robust technique for virtual cutting, Vis Comput 25 (3) (2009) 227–239.
[20] M. Nesme, P.G. Kry, L. Jeřábková, F. Faure, Preserving topology and elasticity for embedded deformable models, in: ACM Transactions on Graphics (TOG), volume 28, ACM, 2009, p. 52.
[21] L. Jeřábková, G. Bousquet, S. Barbier, F. Faure, J. Allard, Volumetric modeling and interactive cutting of deformable bodies, Prog. Biophys. Mol. Biol. 103 (2–3) (2010) 217–224.
[22] C. Dick, J. Georgii, R. Westermann, A hexahedral multigrid approach for simulating cuts in deformable objects, IEEE Trans Vis Comput Graph 17 (11) (2011) 1663–1675.
[23] J. Wu, C. Dick, R. Westermann, Interactive high-resolution boundary surfaces for deformable bodies with changing topology, VRIPHYS 11 (2011) 29–38.
[24] J. Wu, C. Dick, R. Westermann, Efficient collision detection for composite finite element simulation of cuts in deformable bodies, Vis Comput 29 (6–8) (2013) 739–749.
[25] S. Jia, W. Zhang, X. Yu, Z. Pan, Cpu–gpu parallel framework for real-time interactive cutting of adaptive octree-based deformable objects, in: Computer Graphics Forum, volume 37, Wiley Online Library, 2018, pp. 45–59.
[26] Y. Qian, W. Huang, W. Si, X. Liao, Q. Wang, P.-A. Heng, Towards biomechanically and visually plausible volumetric cutting simulation of deformable bodies (2019).
[27] J. Hegemann, C. Jiang, C. Schroeder, J.M. Teran, A level set method for ductile fracture, in: Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation, 2013, pp. 193–201.
[28] J. Wolper, Y. Fang, M. Li, J. Lu, M. Gao, C. Jiang, Cd-mpm: continuum damage material point methods for dynamic fracture animation, ACM Transactions on Graphics (TOG) 38 (4) (2019) 1–15.
[29] Q.Q. Cheng, P.X. Liu, P.H. Lai, Y.N. Zou, An interactive meshless cutting model for nonlinear viscoelastic soft tissue in surgical simulators, IEEE Access 5 (2017) 16359–16371.
[30] S.P. Byeon, D.Y. Lee, Adaptive surface representation based on homogeneous hexahedrons for interactive simulation of soft tissue cutting, Comput Methods Programs Biomed 200 (2021) 105873.
[31] A. Nealen, M. Müller, R. Keiser, E. Boxerman, M. Carlson, Physically based deformable models in computer graphics, in: Computer Graphics Forum, volume 25, Wiley Online Library, 2006, pp. 809–836.
[32] C. Brunel, P. Bénard, G. Guennebaud, P. Barla, A time-independent deformer for elastic-rigid contacts, Proceedings of the ACM on Computer Graphics and Interactive Techniques 3 (1) (2020) 1–21.
[33] G. Allaire, F. Jouve, N. Van Goethem, A level set method for the numerical simulation of damage evolution, in: 6th International Congress on Industrial and Applied Mathematics Zürich, Switzerland, 16–20 July 2007, 2009, pp. 3–22.