IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 18 July 2023, accepted 6 August 2023, date of publication 15 August 2023, date of current version 23 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3305631

RESEARCH ARTICLE

A New Fracture Simulation Algorithm Based On Peridynamics for Brittle Objects

MINGHAO YAN[1] AND DEDAO WU[2]
[1]School of Mathematics and Computer Science, Nanchang University, Nanchang 330031, China
[2]School of Mechanical and Electronic Engineering, Jingdezhen Ceramic University, Jingdezhen 333403, China

Corresponding author: Minghao Yan (minghao.yan@outlook.com)

This work was supported by the National Natural Science Foundation of China under Grant 62262040.

ABSTRACT Fracture simulation can create amazing effects in motion pictures, it is widely used into video games and virtual reality systems. In order to achieve a fast and realistic fracture simulation and in view of the advantages of peridynamics in solving discontinuity problems, we propose a new model, which is based on spring mass model and peridynamics theory, for brittle objects fracture. The method can be described as a two-step strategy. First, the geometric model of an object is preprocessed. The model is completely wrapped in an Axis-Aligned Bounding Box (AABB) envelop box, which is subdivided into multiple fragments with 3d Voronoi diagram. The intersection of fragments and the model surface is computed by using the Boolean algorithm, and the fragments outside the model surface are clipped to obtain the final fragments. The Binary Space Partitioning (BSP) tree subdivide method is adopted in our preprocessing, which further improves the intersection speed. Second, the fracture is calculated according to spring-mass system based on peridynamics. We view each point as a spring node. The seed points are connected according to the rules of the spring-mass model based on peridynamics to form the spring topology. The object fracture is calculated according to the spring-mass model based on peridynamics. To further improve render quality, normal bump texture mapping is employed to render the fracture surface. Experimental results show that the proposed model provides users with improved visual feedback while the computational cost is at the same magnitude of other similar methods. The proposed method is especially suitable for the simulation of brittle object fracturing.

INDEX TERMS Fracture simulation, peridynamics, envelop box, spring topology, fragments.

I. INTRODUCTION

Fracture simulation can create amazing effects in motion pictures, such as crack generation, explosions simulation, crushing simulation etc. It is widely used into video games and virtual reality systems [1], [2], [3]. In recent years, research has been focused on enhancing the effect of simulation [4], [5]. However, its realization has always been a challenging problem. Many researchers have devoted themselves to this research, a number of methods have been proposed to simulate fracture [6], [7], [8], [9], [10].

At present, the simulation methods of the fracture object can be roughly divided into two types: physically based approaches and geometrically based approaches. In geometrically based approaches, only the 3d data models of the virtual objects are partitioned by using preprocessing data methods to simulate fracture or the crushing process of the object under the set conditions is simulated by defining a fixed crushing mode. For instance, Raghavachary et al. [11], [12] used the Voronoi diagram to divide the surface of the initial object into grids, and represent the object breakage and the propagation of cracks by using these grids. The advantage of this scheme is that the generation process of the Voronoi polygon can be preprocessed, and the obtained polygon meshes of the object surface can directly participate in computing online without further subdivision. However, with the number of polygonal meshes in the virtual scene increases, the rendering of all objects will take more time. The solution to this problem is only the local area where fragmentation occurs is calculated, and then

The associate editor coordinating the review of this manuscript and approving it for publication was Lei Wei.

the original object is replaced with a subdivided new object. Mould et al. [13] used an image filter to transform an input line into an image of the fracture surface of an object. Their method is based on the weighted Voronoi diagram. First, the edge of the area covered by the Voronoi diagram is regarded as a crack, and then the set crack is mapped to the cracked 3D object surface, so as to obtain a more realistic crack pattern. This approach, however, does not take into account the possibility of new debris emerging and the new ways by which these fragments move. Oda et al. [14], [15] implemented a multi-stage dynamic segmentation method based on octree, which can effectively reduce the number of tetrahedrons to improve computational efficiency. Based on Havok engine, Eberle [16] simulated the breaking animation of objects with high computational efficiency under local impact. Aurelien Martinet et al. [17] proposed a process-based method to simulate the crushing of solid materials. This method could consume a lot of calculation time and can not control the crack propagation and fracture process well. However, it provides the user with a simple tool that automatically generates many types of cracks and many shapes of debris by adjusting various parameters.

Although the geometrically based methods have higher computational efficiency, the simulation effect is mechanical and lacks authenticity to some extent. The physically based methods are mainly to simulate the process of object breaking by establishing mathematical equations, so that the simulation effect can be as close as possible to the real physical process. For instance, spring mass model, finite element model, meshless local petrov-galerkin, and boundary element method etc. have been applied to the object crushing simulation. In the early stage, Norton et al. [18] published a very important literature on object breakage simulation. They used the simple and convenient particle model of spring to calculate the fracture of an object. The particles in the geometry model of the virtual object are connected by springs, and when a spring is stressed beyond its tolerance, the object could break. In their paper, collision detection and response between multiple fragments are also considered when the simulated object was broken. O'brien's research results [19] are of great significance for the simulation of brittle fracture. In their algorithm, the stress and strain of each mesh are first calculated using the finite element method, which was used to judge whether the object will fracture. Then, the principle of fracture mechanics was adopted to calculate the pattern trend and the extension scope of the brittle object, so as to simulate the animation effect with high fidelity. In order to improve computational efficiency, Parker [20] simplified the finite element method. They designed and implemented a new physical simulation engine Digital Molecular Matter (DMM), which could effectively simulate the object crushing animation. Moreover, they have applied the engine to some popular games. However, as a commercial product, the high price of this engine makes people afraid. Bao's research is also based on the finite element method [21]. The object is divided into tetrahedral mesh volume elements with virtual node method, and then a time-average stress method is used to directly analyze the finite element. Su [22] increases the energy and momentum conservation principle of the object based on Baos method, and processes the collision point to satisfy the real-time simulation.

In addition, meshfree method is also employed to simulate fracture. Pauly et al. [23] represented geometric models of the objects with multiple discrete nodes, which were obtained by simplifying the volume data of geometric models of objects. However, these discrete nodes do not participate in the rendering, and the algorithm only renders the mesh on the object surface. If the stress at the joints of the mesh on the surface of the object is greater than the initial threshold, the object will break. The fragmentation between these discrete points will be propagated by the surface mesh and eventually cause the object to be broken. After the crushing, the cracked units need to be reorganized in the mesh. Simulation using this algorithm can obtain more accurate broken data and more realistic broken animation effect. However, the increase of the number of sampling points will increase the calculation cost of the system, so this algorithm is difficult to satisfy real-time requirements. Zhang et al. [24] combined the finite element method with the meshless method based on point primitives, and the original geometric models in the simulation system were all divided into tetrahedral grids. If the stress level of an individual element exceeded the set limit, the corresponding nodes were transformed into particles with mass. Then the motion of all points is calculated by using the principle of molecular dynamics. The algorithm avoids the problem of mesh reconstruction caused by mesh method and the problem of system stability caused by meshless method when the object is broken, so the simulation results show good visual fidelity. Liu et al. [25] implemented an improved meshless brittle object breakage simulation algorithm. They treat brittle objects as perfectly rigid bodies. They developed a novel model based on damage mechanics for debris generation.

In the models mentioned above, the computation efficiency of geometrically based approaches is high, which can meet real-time virtual simulation. However, in geometrically based approaches, the distribution of fragments and the way objects are broken does not accord with the physical reality. Physically based approaches can usually achieve accurate simulation result, high computation cost is their shortcoming. In this paper, we combine the geometric method with the physical method to proposed a new fracture simulation algorithm based on peridynamics for brittle objects. The method consists of two procedures. First, the geometric models of the virtual objects are divided into multiple fragments using a new three-dimensional Voronoi diagram method, and the seed points regarded as spring nodes in each fragment are connected to form spring topology. Then, the fracture is computed according to spring-mass system based on peridynamics.

This paper is organized as follows. Section II describes the describes the key idea of peridynamics theory. In Section III, We propose the algorithm based on peridynamics, which is

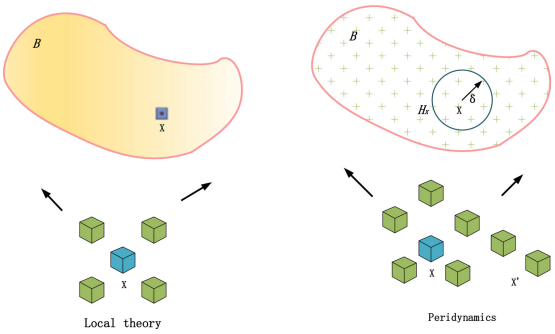


FIGURE 1. Peridynamics model.

able to simulate fracturing for brittle object. Section IV shows that the proposed method provides a superior quality in terms of visual effect compared with classical fracturing based on tetrahedral partition algorithm. The paper is concluded with some suggestions to future work in Section V.

II. PERIDYNAMICS THEORY

Peridynamics is the basic knowledge of our proposed method, which is a theory of continuous mechanics based on nonlocal theory proposed by Silling [26]. As a non-local theory, it is assumed that every material point has a domain centered on itself, all other material points in the domain can interact with this point [27], [28], [29], [30], [31]. This domain is called the near-field range of material points. As shown in Fig. 1, although there is no direct contact between the two points x and x', there is an interaction between two points because point x' is in the near field of point x.

In peridynamic theory, the spatial integral equations are used to describe the motion of material, which are defined at discontinuities. Thus, peridynamics is better suited for modelling crack initiation and propagation and does not need any specific treatment at discontinuities. The original form of the peridynamic model, which for purposes of this paper will be called the bond-based theory, uses a two-particle force function f to describe the interaction between material particles:

$$\rho(x)\ddot{u}(x,t) = \int_{H_x} f(u(x',t) - u(x,t), x' - x)\mathrm{d}V_{x'} + b(x,t) \tag{1}$$

where H_x is a neighborhood of x, u is the displacement vector field, b is a prescribed body force density field, ρ is mass density in the reference configuration, and f is a pairwise force function whose value is the force vector (per unit volume squared) that the particle x' exerts on the particle x. In the following discussion, the relative position of these two particles in the reference configuration will be denoted by ζ and their relative displacement by η:

$$\zeta = x' - x \tag{2}$$

$$\eta = u(x',t) - u(x,t) \tag{3}$$

Using these definitions, $\zeta + \eta$ represents the current relative position vector between the particles. The vector η is called a bond. The physical nature of the interaction between x and x' need not be specified. The concept of a bond that extends over a finite distance is a fundamental difference between the peridynamic theory and the classical theory, which is based on the idea of contact forces (interactions between particles that are in direct contact with each other). Certain restrictions on f arise from basic mechanical considerations:

$$f(-\eta, -\zeta) = -f(\eta, \zeta) \tag{4}$$

$$(\zeta + \eta) \times f(\eta, \zeta) = 0 \quad \forall \eta, \zeta \tag{5}$$

It is convenient to assume that for a given material, there is a positive number δ, called the horizon, such that

$$|\zeta| > \delta \Rightarrow f(\eta, \zeta) = 0 \quad \forall \eta \tag{6}$$

III. PROPOSED METHOD

In order to realize fast and realistic fracture simulation of the brittle object, a new simulation algorithm based on peridynamics is presented. The process of the method can be divided into two parts, which includes preprocessing and algorithm execution. In preprocessing process, the generation and quality calculation of the fragments are main tasks. In the algorithm execution part, the fracture of the brittle object is computed based on peridynamics and the fracture surface is rendered. The details are as follows.

A. A NEW DIVISION METHOD BASED ON THREE-DIMENSIONAL VORONOI DIAGRAM

The division method based on three-dimensional Voronoi diagram has many advantages, such as irregularity, controllability and rapidity. However, for complex models, the ideal fragments cannot be obtained just with Voronoi diagram. To obtain the fragments quickly and accurately, a new algorithm for fragments generation is proposed. First, the virtual object is wrapped in an AABB bounding box, and the seed points are distributed in the bounding box. Then, the bounding box is divided to generate fragments. Finally, the Boolean algorithm is adopted to obtain the relationship between the fragments and the surface of the object, and the final fragments can be achieved by clipping the pieces lying outside the surface of the object.

1) THE DISTRIBUTION OF THE SEED POINTS

In real life, the number and volume of fragments produced for brittle objects shows a certain distribution law. This distribution law is affected by many factors, such as the material of the object, the location of the impact point, the impact velocity and so on. In this paper, the collision point between the colliding object and the brittle object is predicted by the velocity and the initial position of the colliding object, and the number distribution of seed points is determined according to the velocity of the colliding object to simulate the fracture. Our method is based on the following hypothesis: the higher the velocity of the colliding object, the more fragments generated by the collision; and more and smaller pieces are generated in the area closer to the collision point. The virtual

object is first wrapped in an AABB bounding box. According to the distance between the points in the bounding box and the collision point, the seed points were distributed in bounding box according to the normal distribution law to simulate the distribution of fragments.

Algorithm 1 The Division of AABB Bounding Box

input: Geometric model of the object
output: The final fragments of the object
1: Select the seed points
2: Sort the seed points and the Delaunay triangulation is achieve
3: The redundant tetrahedron is removed to obtain the final Delaunay tetrahedron mesh
4: The Voronoi graph is obtained according to the corre-sponding Delaunay tetrahedral mesh
5: The Voronoi diagram outside the bounding box is clipped
6: Obtain the fragments of the AABB bounding box

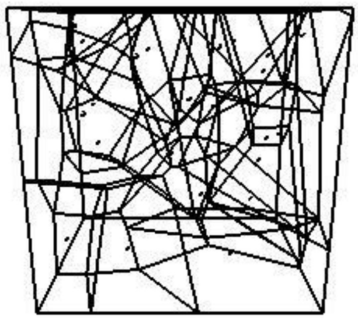


FIGURE 2. The division of the AABB bounding box.

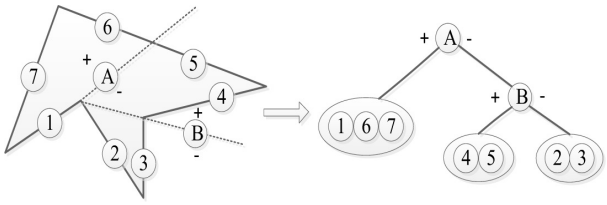


FIGURE 3. Two-dimensional space division based on BSP tree.

2) THE DIVISION OF THE AABB BOUNDING BOX

As shown in Algorithm 1, after the distribution of seed points is determined, seed points are first sorted and the Delaunay triangulation is achieve according to incremental insertion method. Then, a tetrahedron that can contain all the vertices is constructed. By deleting other tetrahedrons associated with the vertex of the tetrahedron, the final Delaunay tetrahedrons are obtained. Finally, the Voronoi diagram is obtained by connecting the center of circumsphere of two adjacent tetrahedrons. And Voronoi diagram outside the bounding box is clipped out, that is, the AABB bounding box of object is divided.

The dividing results are shown in Fig. 2. It can be seen from Fig. 2, the bounding box has been divided into multiple fragments with Voronoi diagram, in which the black points are seed points, and each seed point is combined with its surrounding edges to form a fragment. The algorithm in this section only divides the AABB bounding box, and the multiple resulting fragments cannot reflect the shape of the 3d object model.

3) FRAGMENTS PRODUCTION

To obtain the fragments of brittle objects that are finally used for fracture simulation, Boolean intersection algorithm are used to clipping the bounding box fragment outside the surface of the three-dimensional object, so as to determine the surface fracture boundary of the object and obtain fragments which can reflect the surface information of the original object.

Both surface of fragments produced from bounding box and the original object surface are made of triangle patches. In general, all triangle patches of a virtual object need to be traversed when each fragment is intersected with a three-dimensional object with traditional Boolean operation, which can ensure accurate results. However, its computational expense is very high. To improve the situation, a Boolean intersection algorithm based on BSP tree is proposed. In this method, the object is first divided into several convex sets by BSP tree, and then the intersection between the block model and these convex sets is obtained. The details are as follows:

Space division of BSP tree. The object is divided in space by using BSP tree. As shown in Fig. 3, the two-dimensional space is taken as an example. The non-convex polygons are divided by hyperplanes A and B. The numbers in the figure represent the edges of a non-convex polygon. For instance, the number 1, 2, 3, 4 and 7 are original edges. The number 5 and 6 are new edges generated by the partition of hyperplane A. The partition process will end until the data sets stored in each leaf node are convex set, and the partition result is stored in the files.

Intersection operation. An AABB bounding box is built for each fragment, vertex set of AABB bounding box of fragment i is set $p_i = p_{i_1}, p_{i_2}, \cdots, p_{i_8}$, the positional relationship between p_i and the hyperplane f_k belonged $F = f_1, f_2, \cdots, f_n$ stored in BSP is judged. If p_i and f_k are on the same side, positional relationship continue to be judged until eight vertexes belonged p_i are not on the same side of the hyperplane, which indicates that the bounding box fragment intersects the subtree.

Triangulation was performed on the surface of each bounding box fragment. The intersection of each bounding box fragment and the BSP subtree is obtained by using fast intersection algorithm for spatial triangles. The intersection is the surface model of the fragment required for the final crushing simulation.

The process for Boolean operation is shown in Fig. 4, where the black wireframe for the packaged Bunny model

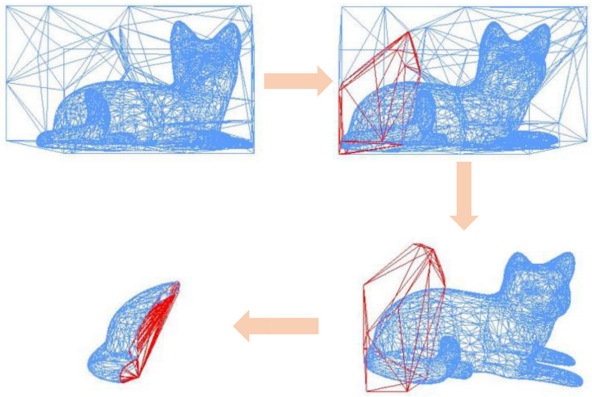


FIGURE 4. The process of Boolean operation for a cat.

is the set of triangulated fragmentation models, and the red wireframe is one of the selected bounding box fragmentation models. The selected bounding box fragmentation model intersects with the Bunny model to obtain the fragmentation model in the lower left corner of Fig. 4. It can be obviously seen that the surface grid information of the cat model is retained very well in the result fragments.

B. THE MASS CALCULATION OF FRAGMENTS

In order to carry out simulation calculation, the fragments need to be given quality attributes. In this paper, the quality of the fragments is computed according to their volume. The resulting fragments are of complex shapes, which may be convex or concave. The surface of a fragment is considered as a piecewise smooth surface and the volume of fragment can be computed according to Gauss's formula. $F =< x, y, z >$ is defined as a vector field, then the Gauss's formula on the simply connected 3d closed region D surrounded by the boundary of the smooth directed surface S can be expressed as

$$\oiint_S (\mathbf{F} \cdot \mathbf{n}) \cdot \mathrm{d}\mathbf{S} = \iiint_D (\nabla \cdot \mathbf{F}) \mathrm{d}V \tag{7}$$

where n is the external normal vector of any point on the surface S. To calculate the volume of a fragment, a vector field F needs to be determined to satisfy $\nabla \cdot F = 1$. In this paper, $F = 1/3 < x, y, z >$. Suppose the fragment P is made up of N triangles, for any triangle $T_i(a_i, b_i, c_i)$ belonged the set of triangles $T = T_0, T_1, \ldots T_{N-1}$, vertex coordinates sorted counterclockwise of T_i are expressed with a_i, b_i, c_i, respectively. The volume of the fragment P can be obtained as

$$V = \iiint_V dV = \frac{1}{3}(\oiint_S < x, y, z > .n)dS \tag{8}$$

$$= \frac{1}{3}\sum_{i=0}^{N-1}\int_{A_i} a_i n_i = \frac{1}{6}\sum_{i=0}^{N-1} a_i \hat{n}_i \tag{9}$$

where n_i is outer normal vector of T_i, $n_i = \hat{n}_i / \left|\hat{n}_i\right|$, and $\hat{n}_i = (b_i - a_i) \otimes (c_i - a_i)$. Its modulus is twice the area of the triangle. Once the volume of the fragments has been determined, the masses of the individual fragments can be easily determined since the objects we study are homogeneous.

C. SPRING-MASS SYSTEM BASED ON PERIDYNAMICS

Peridynamics systems can be characterized as spring-mass systems with two particular properties. First, spring forces are based on a simple strain metric, ε, which normalizes a spring's deformed length by its rest length, thereby decoupling spring stiffness from spring length. Specifically, the traditional spring force exerted on node i by node j is

$$f_i = -k(||x_i - x_j|| - d_{ij})\frac{x_i - x_j}{||x_i - x_j||} \tag{10}$$

where x_i and x_j are the deformed positions of nodes i and j, d_{ij} is the springs rest length, and k is the spring stiffness. Instead, a simple strain metric can be defined in peridynamics

$$\epsilon = \frac{||x_i - x_j|| - d_{ij}}{d_{ij}} = \frac{||x_i - x_j||}{d_{ij}} - 1 \tag{11}$$

and then defines the force as

$$f_i = -k\epsilon\frac{x_i - x_j}{||x_i - x_j||} \tag{12}$$

Note that ϵ is dimensionless. Consequently, springs of different lengths will adopt the same stiffness.

Second, how the springs connect is a important problem in a standard spring-mass system. In general, nodes (point masses) are connected based on a mesh topology, which is defined using 1-ring neighborhoods. In peridynamics systems, spring connections are set between all pairs of nodes which are within a certain distance from one another. Specifically, given a length scale, λ, a spring is placed between any pair of nodes that are within a distance $\delta = N\lambda$ from one other, where N is an integer and its value is between 2 and 6. This will create networks that can include hundreds of springs per node, which is far beyond the connectivity found in typical spring-mass systems.

To simulate fracture, a spring will be removed from the system when its strain exceeds a threshold, τ. Where k and τ depend not only on properties of the material, but vary with δ. Specifically,

$$k = \frac{18k}{\pi\delta^4} \quad and \quad \tau = \sqrt{\frac{5G}{k\delta}} \tag{13}$$

where K represents the bulk modulus and G denotes the fracture energy of the material. Similar to [32], the spring constant is obtained. Note that spring stiffness decreases quickly with larger values of τ. To simulate nonuniformity in solids, one can vary τ of each spring (a normal distribution is used with a standard deviation of 2%). To simulate the phenomenon that some brittle objects become stronger under compression, the effective threshold, τ', can be updated by

$$\tau' = \tau - \alpha min(0, \epsilon_{min}) \tag{14}$$

where α is a constant (we use 0.25) and ϵ_{min} is the minimum strain over the incident springs.

Finally, velocity Verlet is employed. While much maligned for their onerous timestep restrictions, explicit integration is not only simple to implement, but, the small timesteps greatly simplify the simulating of fracture dynamics. With large time steps, one must simulate the fracture propagation and the resulting energy could release which happens over the timestep. In contrast, for small timesteps, fracture propagation can be simulated by simply remove a subset of the active springs every timestep and the resulting energy release is accounted for in the next time step.

In this paper, since every fragment of the object contains a seed point, we regard the seed points as spring nodes, the fragments can be regarded as material block. According to spring-mass system based on peridynamics, the fracture of an object can be computed.

D. THE RENDERING OF FRACTURE PLANE

In general, the fracture surface of brittle object is uneven. To achieve high realistic simulation, normal bump texture mapping is employed in this paper. Compared with traditional concave and convex texture mapping method, normal bump texture mapping has a higher computational efficiency because it can be read directly from the texture coordinates of each pixel of normal vector, which saves a lot of pixel level of normal information.

The principle of normal concave and convex texture mapping is to store the normal vector as color value in a normal map. Although modern shaders can directly import the normal value generated by Perlin noise, the normal map is generated and stored offline to ensure the real-time performance of the system. The RGB component of the color is corresponding to the normal component X,Y and Z. However, the value range of the RGB tri-channel is [0,255], while the value range of the unit normal component is [-1,1]. It is necessary to process the normal value. Specifically, the projection values of the normal in each direction are first transformed into [0,1]. Then, These values are multiplied 255 and converted integer, and finally are stored in the corresponding RGB three color components, respectively.

So far, the grayscale image that stores the height value is converted to a normal map that stores the normal vector. The normal map is used to render the fracture surface.

E. THE IMPLEMENTATION PROCESS OF OUR ALGORITHM

The implementation process of the proposed algorithm is shown in Fig. 5. Firstly, the whole model is wrapped in an AABB envelop box, and seed points needed to generate 3d Voronoi diagram are scattered in the envelop box instead of the virtual object. When the virtual object is subdivided, only the enveloping box is subdivided to generate fragments, and there is no need to detect the complex object surface, which greatly reduces computation time. Then, the Boolean algorithm is used to clip the fragments outside the model surface to obtain the final fragments. Moreover, each fragment is assigned to a certain mass according to its volume, and a constraint is established between two fragments. All the fragments are linked together under this constraint. At last, the force applied to each fragment is computed according to peridynamics model. When the external force on the brittle object is greater than the constraint threshold, the fragments are generated.

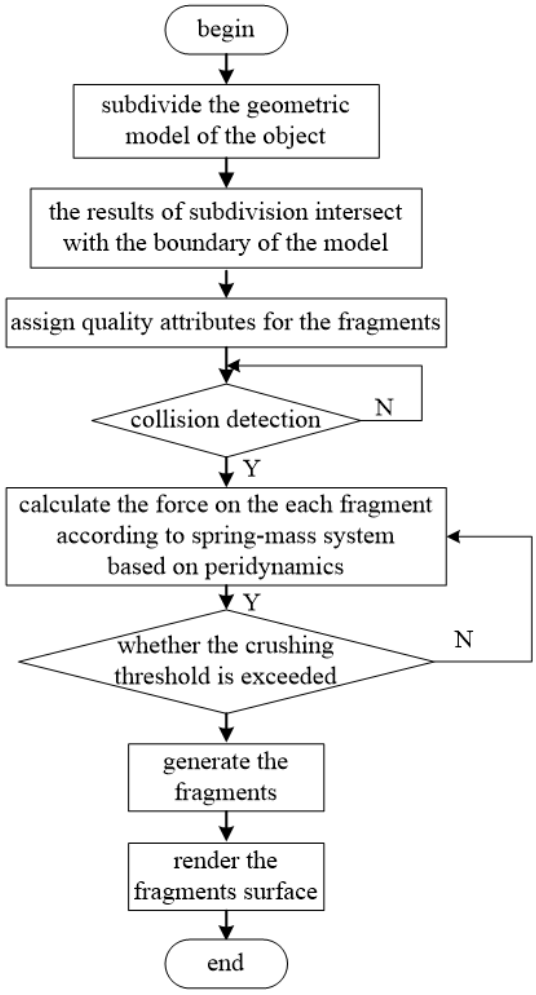


FIGURE 5. The implementation process of our algorithm.

IV. RESULT

The proposed method was implemented in the fracture simulation of various brittle objects. The experimental setup includes a PC with Intel(R) Core(TM) i7-6700 CPU, 8.0 GB RAM and an Intel(R) HD Graphics 530 Graphics card. The program was written with C++ and OpenGL.

A. AUTHENTICITY VERIFICATION

To verify the authenticity of the simulation effect, the fracture simulation based on voronoi graph proposed in this paper and the fracture simulation based on the tetrahedral volume element division method presented in literature [33] were carried on the cube with a simple boundary, respectively. The comparison results are shown in Fig. 6. It can be seen from the left image of Fig. 6 that the shape of the fragments of the cube computed with the method proposed in literature [33]

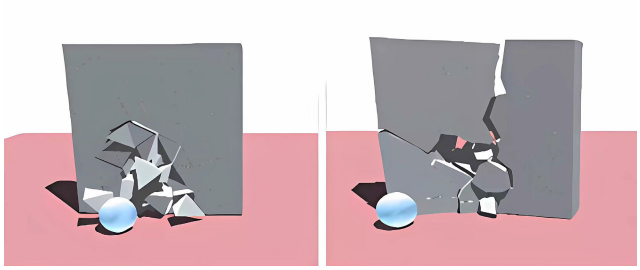


FIGURE 6. Comparison of the visual effect from our algorithm and the method proposed in [33].

TABLE 1. Comparison of the computation time from our method and the method proposed in [33].

method	the number of vertexes	the number of triangles	computation time of a frame(s)
the method in [33]	1792	1792	0.046
our method	1914	3428	0.012

is similar and their size is uniform, which is not consistent with the reality. while the shape of fragments computed by our method (see the right image of Fig. 6) is irregular. Moreover, the closer to the impact point, the more fragments are produced. Otherwise, the further away from the impact point, the less fragments are produced, which is consistent with physical reality. Compared with the traditional crushing method based on tetrahedral volume elements, the fragments produced by the method based on three-dimensional Voronoi diagram are more similar to the real fragments, and the distribution of the fragments' quantity and volume conform to the laws of physics.

Table 1 summarizes the computation time of a frame of the method proposed in [33] and our method. As observed, for a similar number of points and triangles, the computation time of a frame of the proposed method is much fewer than that of the method proposed in [33]. Our method can satisfy the real-time interaction in simulation.

In addition to the calculation of the fragments, the fracture surfaces were also processed in our proposed method. To simulate brittle objects fracturing with high realism, perlin noise interference was applied to rendering of the fracture surfaces. To show the render effects, a cube and a bunny were designed falling freely from a determined height to the ground and cracking, respectively. The number of fragments were set in advance. The experimental results are showed in Fig. 7 and Fig. 8, it can be observed that comparing to the fragments not interfered by noise, the fragments surface interfered by perlin noise is more rough and more realistic.

Table 2 shows the computation time of a frame of the cube and the bunny without the interfered triangles and with the interfered triangles, respectively. It can be seen that the running time of a frame of object with interfered triangles is little more than that without interfered triangles when the number of vertexes, the number of fragments and the number of triangles are same. However, the fracture simulation of two objects satisfy the real-time virtual scene.

TABLE 2. Computation time of cube and bunny without the interfered triangles and with the interfered triangles.

object	the number of vertexes	the number of fragments	the number of triangles	the number of the interfered triangles	computation time/frame(s)
cube	876	50	1568	0	0.007
cube	876	50	1568	1164	0.011
cube	1768	100	3152	0	0.012
cube	1768	100	3152	2388	0.021
bunny	9544	50	19630	0	0.019
bunny	9544	50	19630	5150	0.029
bunny	11925	100	23146	0	0.030
bunny	11925	100	23146	7463	0.043

TABLE 3. Comparison of the fracture computation time from different objects with same height.

object	height(m)	the number of vertexes	the number of triangles	the number of fragments	computation time/frame(s)
bunny	10	7276	14476	26	0.010
horse	10	11209	22304	26	0.016
duck	10	6046	11996	23	0.011
cube	10	256	420	23	0.006
bunny	20	9468	18764	42	0.017
horse	20	12167	24150	44	0.023
duck	20	8254	16132	49	0.024
cube	20	504	856	38	0.011

B. COMPUTATIONAL EFFICIENCY TEST

To test the computational efficiency of the proposed method, four different objects are set to fall freely from the different heights to the ground, and then cracking happen. The simulation effects are shown in Fig. 9. The left images of every row in Fig. 9 show static object, the middle images of every row in Fig. 9 show the fracture effect of object which fall freely from a height of 10 m to the ground and the right images of every row in Fig. 9 show the fracture effect of object which fall freely from a height of 20 m to the ground. It can be obviously observed that the higher an object falls from, the more fragments it produces, and the smaller and more fragments are formed in the part of an object close to the ground, which is consistent with the laws of physics.

Table 3 summarizes the computation time of a frame for four different objects. As can be seen that the higher the height, the more debris is produced for the same object. For the same height and the same number of fragments, the computation time of the object with more complex surface is more than that of the object with simple surface. Furthermore, they all satisfy real-time simulation.

C. APPLICATION OF THE ALGORITHM

The proposed method was employed in an interactive system. Bunny and horse were impacted by a ball with different velocity, which led to the fracture of the object. The fracture effects are shown in Fig. 10. The first column images of Fig. 10 show the fracture effect when the velocity of the ball is 10 m/s. The second column and third column images of Fig. 10 show the fracture effect when the velocity of the ball is 15 m/s and 20 m/s, respectively. It can be seen that the distribution of fragments were basically in accordance with physical laws.

Table 4 summarizes the computation time of a frame for different patterns of fragmentation. It can be observed that the higher the speed of the ball, the greater the impact. And

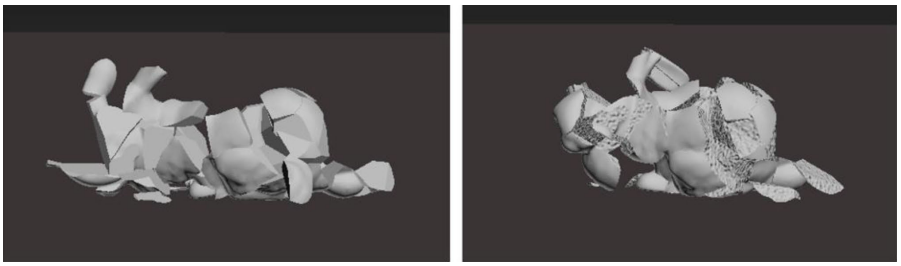


FIGURE 7. The fracturing effect of a bunny. Left: the rendering effect of the fracture surface with traditional method. Right: the rendering effect of the fracture surface with normal bump texture mapping.

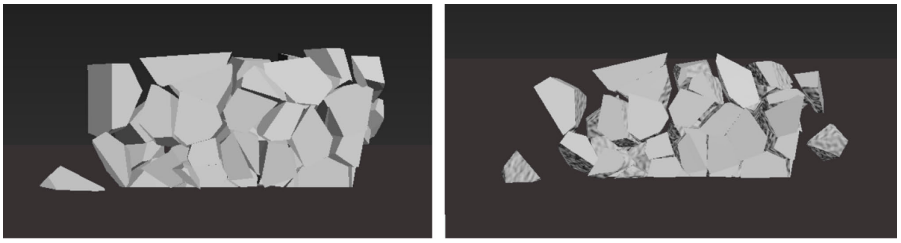


FIGURE 8. The fracturing effect of a cube. Left: the rendering effect of the fracture surface with traditional method. Right: the rendering effect of the fracture surface with normal bump texture mapping.

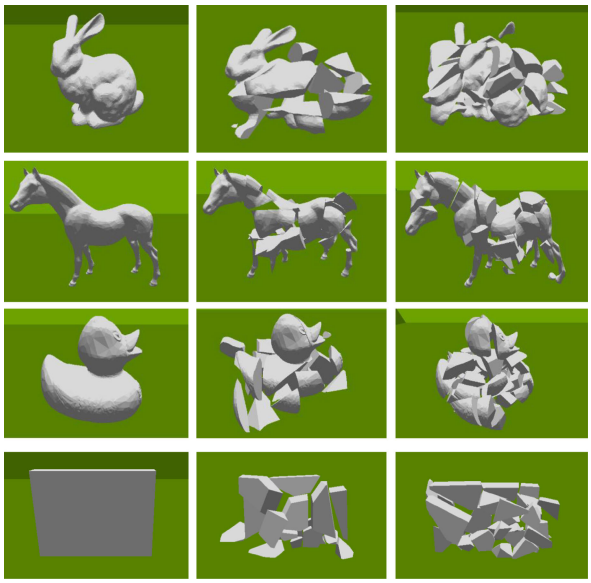


FIGURE 9. The fracturing effect of bunny, horse, duck and cube from different height.

TABLE 4. The computation time of a frame for different patterns of fragmentation.

object	speed of the sphere	the number of vertexes	the number of triangles	the number of fragments	computation time/frame/s
bunny	10	6776	13508	8	0.007
bunny	15	7946	15752	35	0.013
bunny	20	11990	23576	100	0.029
horse	10	9303	18558	13	0.011
horse	15	10683	21270	43	0.019
horse	20	15207	30002	97	0.032

more debris were produced, more time were spent. It can be concluded that the fracture simulation with the proposed method can meet real-time interactive systems.

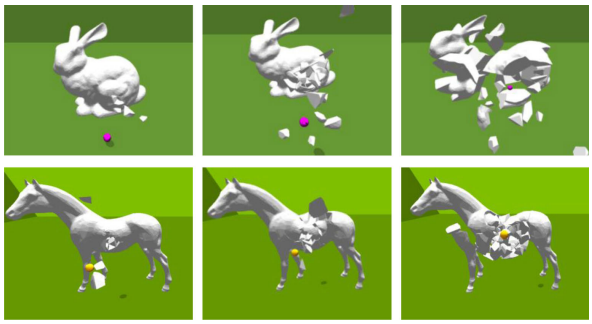


FIGURE 10. The fracture effect of a bunny collided by a ball flying at 10m/s (the left image of the first row), 15m/s (the middle image of the first row) and 20m/s (the right image of the first row) and the fracture effect of a horse collided by a ball flying at 10m/s (the left image of the second row), 15m/s (the middle image of the second row) and 20m/s (the right image of the second row).

V. CONCLUSION

To achieve fast and realistic fracture simulation, we proposed a new fracture simulation algorithm based on peridynamics for brittle objects. In our algorithm, a two-step strategy is employed in this paper. The first step is the geometric model of an object is preprocessed. The second step is the fracture is calculated according to spring mass system based on peridynamics. Finally, to improve render quality, normal bump texture mapping is employed to render the fracture surface. The proposed method was also implemented in the fracture simulation of multiple brittle objects, Compared with other similar algorithms, the proposed algorithm obtains better visual effect with lower computational cost.

In future work we will focus on further algorithm optimizations. The better rendering algorithms would be studied to get a more realistic visual effect of the fracture surface. And

the proposed model will be implemented in a more complex virtual reality system which includes more virtual objects. Because the algorithm has the advantage of high computation efficiency, it can become a preferred method in more virtual system, we will consider to simulate the tearing of soft object using our proposed method.

REFERENCES

[1] S. Sarkar, I. V. Singh, and B. K. Mishra, "A simple and efficient implementation of localizing gradient damage method in COMSOL for fracture simulation," *Eng. Fract. Mech.*, vol. 269, Jun. 2022, Art. no. 108552.

[2] X. Liu, S. Miramini, M. Patel, P. Ebeling, J. Liao, and L. Zhang, "Development of numerical model-based machine learning algorithms for different healing stages of distal radius fracture healing," *Comput. Methods Programs Biomed.*, vol. 233, May 2023, Art. no. 107464.

[3] S. Miramini, Y. Yang, and L. Zhang, "A probabilistic-based approach for computational simulation of bone fracture healing," *Comput. Methods Programs Biomed.*, vol. 180, Oct. 2019, Art. no. 105011.

[4] S. Xia, B. Fu, B. Wang, J. Wu, Y. Cui, and X. Wang, "Computed tomography imaging-based preoperative virtual simulation for calcaneal fractures reduction," *J. Foot Ankle Surg.*, vol. 58, no. 2, pp. 248–252, Mar. 2019.

[5] B. Chen, B. R. Barboza, Y. Sun, J. Bai, H. R. Thomas, M. Dutko, M. Cottrell, and C. Li, "A review of hydraulic fracturing simulation," *Arch. Comput. Methods Eng.*, vol. 29, pp. 1–58, Oct. 2021.

[6] B. Desbenoit, E. Galin, and S. Akkouche, "Modeling cracks and fractures," *Vis. Comput.*, vol. 21, nos. 8–10, pp. 717–726, Sep. 2005.

[7] J. Ning, H. Xu, L. Zeng, and S. Li, "Real-time fracture animation of heterogeneous material on GPU," in *Proc. Comput. Graph. Int.*, Ottawa, ON, Canada, Jun. 2011.

[8] L. Glondu, M. Marchal, and G. Dumont, "Real-time simulation of brittle fracture using modal analysis," *IEEE Trans. Vis. Comput. Graphics*, vol. 19, no. 2, pp. 201–209, Feb. 2013.

[9] J. Barbic and Y. Zhao, "Real-time large-deformation substructuring," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 91:1–91:7, Jul. 2011.

[10] L. Glondu, M. Marchal, and G. Dumont, "Evaluation of physical simulation libraries for haptic rendering of contacts between rigid bodies," in *Proc. ASME World Conf. Innov. Virtual Reality*, Jan. 2010, pp. 41–49.

[11] S. Raghavachary, "Fracture generation on arbitrary surfaces using Voronoi polygons," Dept. CIS, Ohio State Univ., Columbus, OH, USA, Tech. Rep., 1992.

[12] S. Raghavachary, "Fracture generation on polygonal meshes using Voronoi polygons," in *Proc. ACM SIGGRAPH Conf. Abstr. Appl.*, New York, NY, USA, Jul. 2002, p. 187.

[13] D. Mould, "Image-guided fracture," in *Proc. Graph. Interface*. Victoria, BC, Canada: Canadian Information Processing Society, 2005, pp. 219–226.

[14] O. Oda and S. Chenney, "Fast dynamic fracture of brittle objects," in *Proc. ACM SIGGRAPH Poster*, Fremont, CA, USA, 2005, p. 113.

[15] O. Oda and N. Subramaniam, "Fast dynamic fracture of brittle objects in 3D," in *Proc. ACM SIGGRAPH Res. Posters*, New York, NY, USA, 2006, p. 128.

[16] D. Eberle, O. Strunk, and R. O'Sullivan, "A procedural approach to modeling impact damage," in *ACM SIGGRAPH Sketches Appl.*, Fremont, CA, USA, 2003, p. 1.

[17] A. Martinet, E. Galin, B. Desbenoit, and S. Hakkouche, "Procedural modeling of cracks and fractures," in *Proc. Shape Modeling Int.*, 2004, vol. 21, no. 8, pp. 346–349.

[18] A. Norton, G. Turk, B. Bacon, J. Gerth, and P. Sweeney, "Animation of fracture by physical modeling," *Vis. Comput.*, vol. 7, no. 4, pp. 210–219, Jul. 1991.

[19] J. F. O'Brien and J. K. Hodgins, "Graphical modeling and animation of brittle fracture," in *Proc. 26th Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, New York, NY, USA, 1999, pp. 137–146.

[20] E. G. Parker and J. F. O'Brien, "Real-time deformation and fracture in a game environment," in *Proc. ACM SIGGRAPH/Eurograph. Symp. Comput. Animation*, Metairie, LA, USA, 2009, pp. 165–175.

[21] Z. Bao, J.-M. Hong, J. Teran, and R. Fedkiw, "Fracturing rigid materials," *IEEE Trans. Vis. Comput. Graphics*, vol. 13, no. 2, pp. 370–378, Mar. 2007.

[22] J. Su, C. Schroeder, and R. Fedkiw, "Energy stability and fracture for frame rate rigid body simulations," in *Proc. ACM SIGGRAPH/Eurograph. Symp. Comput. Animation*, Metairie, LA, USA, Aug. 2009, pp. 155–164.

[23] M. Pauly, R. Keiser, B. Adams, P. Dutré, M. Gross, and L. J. Guibas, "Meshless animation of fracturing solids," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 957–964, Jul. 2005.

[24] N. Zhang, X. Zhou, D. Sha, X. Yuan, K. Tamma, and B. Chen, "Integrating mesh and meshfree methods for physics-based fracture and debris cloud simulation," in *Eurograph. Symp. Point-Based Graph.*, New York, NY, USA, 2006, pp. 145–154.

[25] N. Liu, X. He, S. Li, and G. Wang, "Meshless simulation of brittle fracture," *Comput. Animation Virtual Worlds*, vol. 22, nos. 2–3, pp. 115–124, Apr. 2011.

[26] S. A. Silling, M. Epton, O. Weckner, J. Xu, and E. Askari, "Peridynamic states and constitutive modeling," *J. Elasticity*, vol. 88, no. 2, pp. 151–184, Aug. 2007.

[27] H. Yao, T. Liang, Y. Xu, G. Hou, L. Lv, and J. Zhang, "Sinus tarsi approach versus extensile lateral approach for displaced intra-articular calcaneal fracture: A meta-analysis of current evidence base," *J. Orthopaedic Surg. Res.*, vol. 12, no. 1, p. 43, Dec. 2017.

[28] J. Ding, Z. Zhang, F. Yang, Y. Zhao, and X. Ge, "Plastic fracture simulation by using discretized virtual internal bond," *Eng. Fract. Mech.*, vol. 178, pp. 169–183, Jun. 2017.

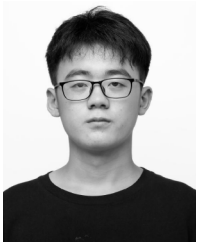
[29] H. Zhang and P. Qiao, "A coupled peridynamic strength and fracture criterion for open-hole failure analysis of plates under tensile load," *Eng. Fract. Mech.*, vol. 204, pp. 103–118, Dec. 2018.

[30] Y. Zhang and P. Qiao, "An axisymmetric ordinary state-based peridynamic model for linear elastic solids," *Comput. Methods Appl. Mech. Eng.*, vol. 341, pp. 517–550, Nov. 2018.

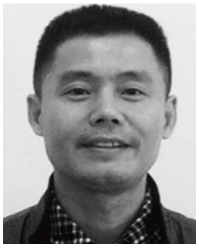
[31] B. Kilic and E. Madenci, "An adaptive dynamic relaxation method for quasi-static simulations using the peridynamic theory," *Theor. Appl. Fract. Mech.*, vol. 53, no. 3, pp. 194–204, Jun. 2010.

[32] S. A. Silling and E. Askari, "Peridynamic modeling of impact damage," in *Proc. Problems Involving Thermal-Hydraulics, Liq. Sloshing, Extreme Loads Struct.* New York, NY, USA: American Society of Mechanical Engineers, 2004, pp. 197–205.

[33] L. Zeng, Y. G. Wu, and S. K. Li, "Real-time simulation of rigid body crushing effects," *Comput. Res. Develop.*, vol. 47, no. 6, pp. 1032–1037, 2011.



MINGHAO YAN is currently pursuing the bachelor's degree in computer science with Nanchang University, Jiangxi, China. He has been delving into computer physics simulation, since 2022. His research interests include deep learning and computer simulation.



DEDAO WU received the Ph.D. degree from Nanchang University, China, in 2021. He has been with Jingdezhen Ceramic University, Jiangxi, China, since July 2006. His research interests include virtual surgery, image processing, and biomechanical calculation.