Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/results-in-physics

Animation of Crack Propagation by means of an Extended Multi-body Solver for the Material Point Method

Joel Wretborn[a,*], Rickard Armiento[b], Ken Museth[a]

[a]DreamWorks Animation, Glendale, CA 91201, USA
[b]Department of Physics, Chemistry and Biology (IFM), Linköping University, SE-581 83 Linköping, Sweden

ARTICLE INFO

Article history:
Received October 25, 2017

Keywords: Material Point Method, cracks, contact

ABSTRACT

We propose a multi-body solver that extends the Material Point Method (MPM) to simulate cracks in computer animation. We define cracks as the intersection between pieces of bodies created by a pre-fracture process and held together by massless particle constraints (glue particles). These pieces are simulated using a MPM multi-body solver extended by us to efficiently handle N-body collisions. Benefits of the present work include (1) low computational overhead compared to a normal MPM algorithm; (2) good scaling in three dimensions due to our use of sparse grids for background computations; (3) allowing for an easy and controllable setup phase to simulate a desired material failure mode, which is especially useful for computer animation.

1. Introduction

Some of the most interesting natural phenomena involve material fracture, and it is a vital ingredient in simulations where realism is desired. Hence, algorithms for object breakage using various simulation techniques are a topic of high level of interest, both for engineering applications and for computer graphics and animation. Specifically for simulations using the material point method (MPM) by Sulsky et al. (1995), simulation of fracture via crack propagation appear to have been mostly discussed in the engineering literature with a focus on numerical accuracy. The aim of these works is different from what is needed for animation applications, where simulation speed and art directability are prioritized. In the present paper we present an algorithm for fracture that provides attractive features for use in computer graphics while only adding a small overhead over regular MPM simulation.

The MPM method is increasingly relevant for simulations of materials due to improvements in hardware and algorithms. It has proven useful in simulations involving large deformations, where the approach of combining meshless particles with a fixed computational background grid provides a robust framework. Both viscoelastic and viscoplastic materials have been simulated with impressive results. The MPM has also been used for other materials like rubber and sponges, which can undergo large elastic deformations. Inherent to the method is that these materials will break naturally if the stress is too high at any particular location. Normally, a simulated material is homogeneous and isotropic. This is often not the case for their real-world counterpart, as small weaknesses and local inconsistencies are important features for how a crack propagates through a medium. Such irregularities could be introduced in the simulation by modifying the parameters that govern the constitutive model on a per-particle basis or by jittering the particles in their initial configuration, but doing so in a way that both conserves the original collective behaviour of the material while achieving the desired break point is difficult.

In the present work, we extend MPM by defining a crack via pre-fracturing of a specimen into different bodies, which are bound together by particle constraints scattered on the crack surface. We call these particles *glue particles*, and their role is to hold the object fragments connected until they break and

*Corresponding author: Tel.: +64 021 023 53668; e-mail: joel@wbn.se;

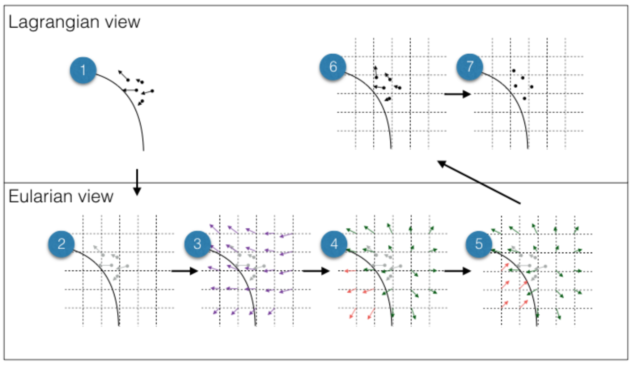

Fig. 1. ***Overview of the algorithm.*** **The steps are classified as *Lagrangian* and *Eulerian* to signify what entity is being manipulated, particles or grid nodes. Explanation of the steps: (1) initialized particles are used as input; (2) a background grid is created; (3) mass and velocity are rasterized onto the grid, and internal forces are calculated using the constitutive model; (4) new velocities are calculated using external and internal forces (red values are inside a boundary); (5) boundary collisions are resolved; (6) velocity is transferred back to the particles; (7) particle position and deformation are updated.**

a crack is formed. The focus of this paper is on simulation of bodies with a single crack, and where the crack propagation is dominated by an opening mode. The pieces from a fractured body are allowed to interact freely in the simulation, and we also present an extension to the contact algorithm by Huang et al. (2011) to allow for arbitrarily many colliding bodies in the same solver.

The rest of the paper is organized as follows. A review of related works is discussed in 2. In section 3 we present the extended contact algorithm, which is utilized for the crack algorithm in section 4. Simulations based on the two algorithms will be shown in section 5, followed by a discussion in section 6 that points out current artefacts and limitations. Final conclusions then follow in section 7.

2. Related work

Early works on the simulation of deformable plasticity and fracture in computer graphics were undertaken by Terzopoulos and Fleischer (1988). Such approaches to dynamic fracture propagation often involved mesh-based finite element methods due to the ease of calculating stress coefficients along connected points. O'Brien and Hodgins (1999) introduced an element splitting approach to increase numerical accuracy and avoid visible artefacts for brittle fracture, which was later extended by O'Brien et al. (2002) to include ductile fracture. However, mesh based methods can have problems handling large deformations, which may easily occur in fracture scenarios due to high internal stresses needed for a crack to surface. Pauly et al. (2005) suggested a meshfree method where the surface of a material is modeled using unconnected points. A crack is explicitly represented by a crack front of surface particles, which are added continuously to the crack front during the simulation. Kaufmann et al. (2009) proposed a method for fracture of thin sheets that requires a pre-defined crack. Explicitly declared cracks are flexible and give great control over how the material breaks. However, visible artefacts will arise if stresses are not properly aligned to the fracture surface, as the resulting crack will look unrealistic. Müller et al. (2013) shows great results combining a pre-computed compound mesh dynamically applied based on the impact location of a projectile, where resulting pieces are simulated by a rigid-body solver.

The MPM was created by Sulsky et al. (1995), and has since then proven to be useful for a range of different phenomena. It was later introduced to computer graphics by Stomakhin et al. (2013) with their work on snow. Ram et al. (2015) and Yue et al. (2015) modified their method to simulate viscoplastic materials like foam. Stomakhin et al. (2014) added a heat solver to simulate phase change of materials. Klar et al. (2016) and Daviet and Bertails-Descoubes (2016) used MPM together with a Drucker-Prager plasticiy model to simulate sand and other granular materials. Yue et al. (2015) complemented MPM with a particle re-sampling scheme to handle potential non-uniform particle distributions due to high shearing strain. Jiang et al. (2015) proposed to track a locally affine transformation on each particle that would enable conservation of angular momentum; an improvement over the normally used PIC/FLIP [Zhu and Bridson (2005)] update scheme.

In MPM, all particles discretized to the same grid will share the same description of internal stresses on the Eularian grid. This will yield non-physical contact forces when two distinct bodies collide, but this can be avoided by complementing MPM with a contact algorithm. Due to the particles in MPM being meshless, contact is often resolved on the grid, and any Eularian contact method can be used. Levin et al. (2011) resolves contact of overlapping Eularian grids by formulating the problem by the principle of least constraints. A similar approach was employed by Fan et al. (2013), who uses this Eularian formulation to simulate contact for a Lagrangian mesh. Huang et al. (2011) proposes a method targeted at MPM. They discretize each body to separate background grids, and an impenetrability condition is imposed with respect to the relative grid velocity to resolve collisions. Hegemann et al. (2013) uses this contact scheme on a purely grid based level set method. However, due to the grid being a smeared representation of the current particle configuration, all purely Eularian methods will observe that collisions are detected prematurely. This artefact is discussed in the context of our solver in section 6.

One existing method to handle the lack of an inherent way to represent cracks in MPM is CRAMP (CRAcks with Material Points) [Nairn (2003); Guo and Nairn (2006)]. CRAMP introduces cracks on the Eulerian grid by allowing grid nodes to have multiple velocity fields. Particles from opposite sides of a crack are rasterized to different grids, which is determined by a line-crossing algorithm from the particle to the grid. The crack is explicitly represented as a Lagrangian mesh of massless particles, which in 2D constitutes of connected line segments and in 3D a polygon mesh. CRAMP is primarily used for engineering applications to investigate the stress response of a specimen with a non-propagating crack, but was recently extended by Bardenhagen et al. (2011) to allow for dynamically propagating cracks. The authors, however, also say that their dynamic crack propagation algorithm "... *[requires] sub-*

stantial computational effort even for two-dimensional calculations." The importance of speed and art directability makes CRAMP difficult to use in graphics applications. Our goal is a crack algorithm capable of simulating realistic looking crack propagation, with lower computational requirements more suitable for computer graphics. Daphalapurkar et al. (2007) have also developed a scheme for crack growth in generalized interpolation MPM (GIMP) which targets engineering applications. They simulate a crack along a pre-defined cohesive zone in a 2D specimen, where particles from opposite sides of the crack interface interact. The glue particles presented in this paper resemble their use of a cohesive zone.

3. Multi-body solver for MPM

MPM is a hybrid method in that it combines an Eulerian mesh with Lagrangian particles. First, a continuous material is discretized into material points. The particles store all information that will be carried on through the simulation such as, position, velocity, deformation, and other potential properties related to the constitutive model. The Eulerian grid is used in the background to perform certain types of calculations. A particle is rasterized onto the grid by means of a weighting function, which transfers its attributes to the grid nodes. The internal and external forces are solved on the grid, and the attributes are transferred back to the particles and their positions are updated. Afterwards, the grid is discarded and a new simulation step is initialized. An overview of the algorithm can be seen in fig. 1. We follow closely the implementation by Stomakhin et al. (2013), with the exception that we use an explicit time step integration scheme to simplify the grid update.

Our approach to crack propagation is to pre-fracture a specimen into separate bodies that are held together by glue particles. When these particle constraints break, each body must interact with every other body in the simulation. Section 3.1 first describes a two-body collision scheme founded upon the work of Huang et al. (2011). This two-body algorithm is a reformulation of the same method they present in their paper, but we introduce a different notation which we use for our extension to N-body collisions. The difference stems from the way we handle surface normals. A normal is typically calculated by a finite difference of the mass on the background grid, and forces are distributed along these normals when bodies collide. For two bodies, conservation of momentum can only occur when the surface normals are collinear. In general this is not the case, and Huang et al. (2011) suggest multiple methods to ensure that this condition is kept. However, when increasing the number of colliding bodies to more than two, we were not able to adapt their algorithm to keep momentum conserved (as collinearity does not scale to more than two vectors). We propose instead to iteratively solve 2-body collision problems until all pair collisions have been resolved. See fig. 2. This allows for arbitrarily many colliding bodies to interact in the same solver, while still guaranteeing conservation of momentum. In section 3.2 we describe the required changes from the 2-body case to the N-body case.

3.1. Two-body systems

To avoid loss of information at the collision interface, each body b will be rasterized to its own set of grids, storing mass m^k_{bi}, force $\boldsymbol{f}^k_{bi}$, and velocity $\boldsymbol{v}^k_{bi}$ and $\boldsymbol{v}^{k+}_{bi}$, for each grid node $\boldsymbol{i}$ at time k. We use a combined PIC/FLIP scheme to transfer grid attributes back to the particles [Zhu and Bridson (2005)], which requires us to store on the grid both initial velocity $\boldsymbol{v}^k_{bi}$, and updated velocity $\boldsymbol{v}^{k+}_{bi}$ after grid forces have been applied. We use the PIC/FLIP blend ratio $\alpha = 0.95$ for all our simulations (see e.g., Stomakhin et al. (2013) for an explanation of α). The grids must be aligned to allow for comparisons between grid values of different bodies, i.e., m^k_{bi} and m^k_{di} must refer to the masses of body b and d respectively at the same world space coordinate. To avoid confusion, we will from here on refer to a *grid node* as a node corresponding to a grid position $\boldsymbol{x_i}$. That means that m^k_{bi} and m^k_{di} contribute to the *same* grid node, but to different grids.

Two colliding grids at a grid node $\boldsymbol{i}$ are required to fulfill the *impenetrability condition*

$$\boldsymbol{v}_{bi} \cdot \boldsymbol{n}_{(b,d)i} + \boldsymbol{v}_{di} \cdot \boldsymbol{n}_{(d,b)i} = 0 \tag{1}$$

where $\boldsymbol{n}_{(b,d)i}$ are outwards pointing normals from b to d and vice versa. To ensure conservation of momentum we require the normals to be collinear, namely

$$\boldsymbol{n}_{(b,d)i} = -\boldsymbol{n}_{(d,b)i}. \tag{2}$$

Combining eq. 2 and eq. 1 yields

$$0 = (\boldsymbol{v}_{bi} - \boldsymbol{v}^{cm}_{\boldsymbol{i}}) \cdot \boldsymbol{n}_{(b,d)}, \tag{3}$$

where we use the center of mass velocity $\boldsymbol{v}^{cm}_{\boldsymbol{i}} = \sum_b m_{bi}\boldsymbol{v}_{bi} / \sum_b m_{bi}$ as in Huang et al. (2011). Thus, two grids are considered to be colliding at a grid node if

$$(\boldsymbol{v}_{bi} - \boldsymbol{v}^{cm}_{\boldsymbol{i}}) \cdot \boldsymbol{n}_{(b,d)} > 0 \iff (\boldsymbol{v}_{di} - \boldsymbol{v}^{cm}_{\boldsymbol{i}}) \cdot \boldsymbol{n}_{(d,b)} > 0. \tag{4}$$

The normal $\boldsymbol{n}_{bi}$ must be derived from the current configuration of the body. We will derive it from the mass grid as

$$\boldsymbol{n}_{bi} = -\frac{\nabla m_{bi}}{|\nabla m_{bi}|}, \tag{5}$$

where the minus sign was introduced to create an outwards facing normal. In general, letting $\boldsymbol{n}_{(b,d)i} = \boldsymbol{n}_{bi}$ and $\boldsymbol{n}_{(d,b)i} = \boldsymbol{n}_{di}$ will not satisfy eq. 2 (see fig. 2). There are multiple methods proposed in Huang et al. (2011) to ensure collinearity. All simulations in this paper use the method of averaging normals with

$$\boldsymbol{n}_{(b,d)i} = -\boldsymbol{n}_{(d,b)i} = \frac{\boldsymbol{n}_{bi} - \boldsymbol{n}_{di}}{|\boldsymbol{n}_{bi} - \boldsymbol{n}_{di}|}. \tag{6}$$

3.1.1. Resolving collisions

Contact is said to occur if eq. 4 is satisfied for any of the bodies using the next nodal velocities $\boldsymbol{v}^{k+}_{bi}$ or $\boldsymbol{v}^{k+}_{di}$. The superscript will be omitted in the following derivation for brevity. If bodies are in contact at a grid node $\boldsymbol{i}$, collision is resolved by applying

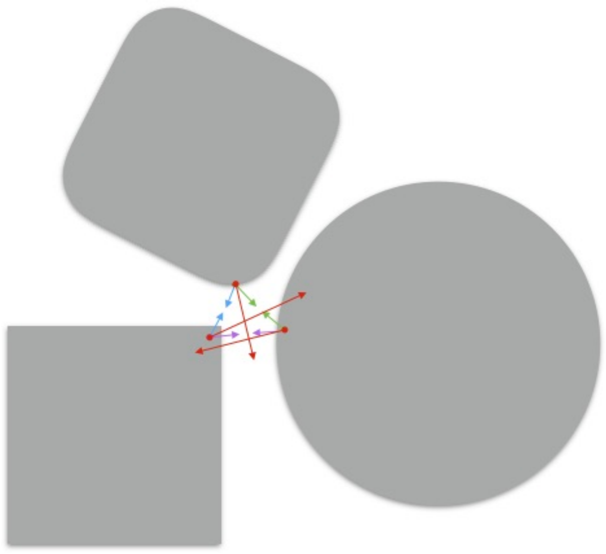

Fig. 2. ***Contact nodes.*** **Three bodies in contact, and separated for clarity. The red vectors are normals calculated from a finite difference on the grid. Recalculated pairwise vectors using eq. 6 are shown from each body, along which contact forces are distributed.**

the impulse $\boldsymbol{f}^*_{bi}$ and $\boldsymbol{f}^*_{di}$ to the respective bodies. By virtue of Newton's third law

$$\boldsymbol{f}^*_{bi} + \boldsymbol{f}^*_{di} = \boldsymbol{0}. \tag{7}$$

The impulse is employed in the direction of the normal by $\boldsymbol{f}^*_{bi} = f^*_b \boldsymbol{n}_{(b,d)i}$ and it is added to the grid force, $\bar{\boldsymbol{f}}_{bi} = \boldsymbol{f}_{bi} + \boldsymbol{f}^*_{bi}$. The purpose of the impulse force is to calculate a correction velocity $\boldsymbol{v}^*_{bi}$ such that the grid node velocity satisfies eq. 1. The velocity relates to the force as

$$\boldsymbol{v}^*_{bi} = \frac{\Delta t}{m_{bi}} \boldsymbol{f}^*_{bi}, \tag{8}$$

and we update the velocity by $\bar{\boldsymbol{v}}_{bi} = \boldsymbol{v}_{bi} + \boldsymbol{v}^*_{bi}$. Using the impenetrability condition in eq. 1, with the corrected velocities $\bar{\boldsymbol{v}}_{bi}$ and $\bar{\boldsymbol{v}}_{di}$, it is possible to calculate the impulse required to deflect the bodies as,

$$f^{norm}_{bi} \equiv \boldsymbol{f}^*_{bi} \cdot \boldsymbol{n}_{(b,d)i} = \frac{m_{bi} m_{di}}{(m_{bi} + m_{di})\Delta t} (\boldsymbol{v}_{di} - \boldsymbol{v}_{bi}) \cdot \boldsymbol{n}_{(b,d)i}. \tag{9}$$

If no friction is desired the collision algorithm is completed. However, the additional work to include friction follows steps very similar to the ones above. Let the tangential unit vector in the direction of the relative velocity at the contact node be $\boldsymbol{s}_{(b,d)i}$ and $\boldsymbol{s}_{(b,d)i} = -\boldsymbol{s}_{(d,b)i}$. Then, the friction equivalent of eq. 9 is

$$f^{tan}_{bi} \equiv \boldsymbol{f}^*_{bi} \cdot \boldsymbol{s}_{(b,d)i} = \frac{m_{bi} m_{di}}{(m_{bi} + m_{di})\Delta t} (\boldsymbol{v}_{di} - \boldsymbol{v}_{bi}) \cdot \boldsymbol{s}_{(b,d)i}. \tag{10}$$

Using the Coulumb friction law, the resulting expression for $\boldsymbol{f}^*_{bi}$ reads as

$$\boldsymbol{f}^*_{bi} = f^{norm}_{bi} \boldsymbol{n}_{(b,d)i} + \min(\mu f^{norm}_{bi}, f^{tan}_{bi}) \boldsymbol{s}_{(b,d)i}. \tag{11}$$

The velocity is now calculated by eq. 8. The most efficient way to compute $\boldsymbol{f}^*_{di}$ is by means of eq. 7. This concludes the contact algorithm.

3.2. N-body systems

The core idea for our treatment of N-body collisions is to introduce pairwise comparisons between bodies. This is straightforward using the notation we introduced in the last section, as the initial assumptions of impenetrability and collinearity still hold. However, due to the additional bodies some calculations are not valid. Most notably,

$$(\boldsymbol{v}_{bi} - \boldsymbol{v}_{di}) \cdot \boldsymbol{n}_{(b,d)i} \neq (\boldsymbol{v}_{bi} - \boldsymbol{v}^{cm}_i) \cdot \boldsymbol{n}_{(b,d)i}, \tag{12}$$

and as a result

$$(\boldsymbol{v}_{bi} - \boldsymbol{v}^{cm}_i) \cdot \boldsymbol{n}_{(b,d)i} \neq (\boldsymbol{v}_{di} - \boldsymbol{v}^{cm}_i) \cdot \boldsymbol{n}_{(d,b)i}, \tag{13}$$

which means that the condition eq. 4 is no longer valid. As before, the velocity of the node is defined as the center of mass velocity, and the velocity of b should be changed if it would penetrate d with respect to v^{cm}_i. Consequently we require for colliding nodes that

$$(\boldsymbol{v}_{bi} - \boldsymbol{v}^{cm}_i) \cdot \boldsymbol{n}_{(b,d)i} > 0. \tag{14}$$

Equation 14 is a generalized statement of eq. 4 despite their similarities, as the latter assumes only two bodies. Now there might exist cases where

$$\begin{aligned} (\boldsymbol{v}_{bi} - \boldsymbol{v}^{cm}_i) \cdot \boldsymbol{n}_{(b,d)i} &> 0, \\ (\boldsymbol{v}_{di} - \boldsymbol{v}^{cm}_i) \cdot \boldsymbol{n}_{(d,b)i} &< 0 \end{aligned} \tag{15}$$

and it is unclear whether any impulse forces should be applied or not. To ensure that eq. 14 is being correctly identified for each pair, we state as a definition that two bodies b, d are in contact if b is in contact with d *or* d is in contact with b. That is, two bodies are in contact if

$$\max((\boldsymbol{v}_{bi} - \boldsymbol{v}^{cm}_i) \cdot \boldsymbol{n}_{(b,d)i}, (\boldsymbol{v}_{di} - \boldsymbol{v}^{cm}_i) \cdot \boldsymbol{n}_{(d,b)i}) > 0. \tag{16}$$

3.2.1. Resolving collisions

As previously, contact is determined using the next nodal velocity $\boldsymbol{v}^{k+}_{bi}$ but with eq. 16 instead of eq. 4. The total impulse for a body b is the sum of all pairwise impulses

$$\boldsymbol{f}^*_{bi} = \sum_{d \neq b} \boldsymbol{f}^*_{(b,d)i}. \tag{17}$$

Now, $\boldsymbol{f}^*_{bi} \neq -\boldsymbol{f}^*_{di}$ in general. Newton's third law states that

$$\boldsymbol{0} = \sum_b \boldsymbol{f}^*_{bi} = \sum_b \sum_{d \neq b} \boldsymbol{f}^*_{(b,d)i}. \tag{18}$$

Solving this equation for 2-body collisions has already been done, as for two bodies eq. 18 turns into the previous eq. 7. For more than two bodies the system is degenerate and multiple solutions that fulfill eq. 18 exist. Posing this as a minimization problem of $\sum_b |\boldsymbol{f}^*_{bi}|$ given eq. 18 and $\boldsymbol{f}^*_{(b,d)i} \cdot \boldsymbol{n}_{(b,d)i} \leq 0$, we ensure that the collision is resolved without interpenetration of objects and while minimizing the virtual work done by the multi-body solver . A similar optimization problem is solved by Fan et al. (2013). We simplify the problem, as mentioned, by doing pairwise collision comparisons and impulse adjustments independently of each other. Mathematically, this is an approximation that corresponds to treating all pairwise contact normals for a grid node as if they were orthogonal to each other. This, in practice, is rare, however, the simplification speeds up computation and does not create any apparent visual artefacts. As a result, the algorithm for 2-body collisions can be repeatedly applied for all pairs at a grid node until all pairs have been resolved. This extension is straightforward given the previous 2-body scheme, and only a few steps need to be revised. The full N-body collisions algorithm is presented in the list below.

1. Rasterize each particle group to a distinct grid.
2. For each grid, calculate $m_i, \boldsymbol{f}_i, \boldsymbol{v}_i$, and $\boldsymbol{v}_i^+$ by the normal MPM algorithm.
3. Locate potential collision nodes, where $m_i > 0$ for more than one grid.
   For each potential collision node $\boldsymbol{i}$*:*
4. Calculate all body normals $\boldsymbol{n}_{bi}$ by eq. 5.
5. Determine if grids are colliding by eq. 16.
   For each grid pair at a collision node:
6. Make normals collinear by eq. 6.
7. Calculate correction forces along pair normals by eq. 11.
8. Apply correction forces to grid pair.

Afterwards, particles are updated from the grids by the normal MPM algorithm.

4. Cracks with the Material Point Method

MPM is a versatile method, and by exerting increasing amounts of stress to a body it will eventually break without special treatment. However, after the material has cracked the different pieces still belong to the same body and will be rasterized to the same grid, and the result will be visible artefacts due to non-physical interaction forces. For example, the bodies may merge together again if kept in contact. Additionally, art directability of a crack is hard and cumbersome. Weaknesses can be introduced by modifying the material properties at places, but these are often unreliable and do not look realistic.

In this work we present a simple but novel algorithm for simulating cracks with MPM. Our approach consists of the following steps. (1) Pre-facture the simulated object by splitting it into different pieces, (2) glue the pieces together using massless material points scattered at the interface between pieces by defining a particle based criterion for when pieces should break apart, and (3) use the multi-body solver to simulate the interaction between separated pieces. These proposed steps are outlined in sections 4.1-4.4, and an algorithmic overview is provided in section 4.5.

4.1. Pre-fracture

The pre-fracturing stage provides the possibility to directly manipulate when and where a certain material breaks. If we split the initial body into different pieces we have created a discontinuity at the surface interface where they meet, and a crack is defined. Figure 3 shows an example of a possible discretization.

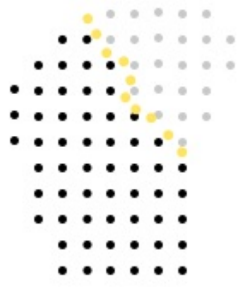
Fig. 3. ***Split.*** **Glue particles (yellow) are scattered at the crack interface.**

4.2. Merging grids

By virtue of the fact that particles are split into separate groups, they will each be rasterized to their own grids. The different pieces, as created by the pre-fracturing process, will thus be seen as separate objects by the multi-body solver, and hence the solver will apply the collision scheme to resolve contact. This is desired in cases where the material have been broken apart, but initially we want to consider the pieces as a single body. This is done by merging the two grids, in the sense that the grids behave as if the particles of each body has been rasterized to the same grid. This can be done by introducing the following operations;

$$\begin{aligned} m_{\text{JOIN}i} &= m_{bi} + m_{di}, \\ \boldsymbol{f}_{\text{JOIN}i} &= \boldsymbol{f}_{bi} + \boldsymbol{f}_{di}, \\ \boldsymbol{v}_{\text{JOIN}i} &= \frac{m_{bi}\boldsymbol{v}_{bi} + m_{di}\boldsymbol{v}_{di}}{m_{bi} + m_{di}}. \end{aligned} \tag{19}$$

The joint grid now represents the combined body of b and d.

4.3. Glue particles

In order to manage a dynamic fracture scenario it is necessary to encapsulate localized information on whether two grids are connected (and should be merged) or separate (and should be treated as such). We introduce a new set of particles G to all particles P, such that $G \subset P$. We denote these particles *glue particles* and will reference them by a running index g. A glue particle

- is massless, and will thus transform with the body while not affecting any grid calculations,
- contains a criterion c_g for the material failure mode. The criterion is a value threshold, and the value will need to be transferred to the grid,
- has a radius of influence in which it will affect grid nodes (and will *glue* grids together by merging them using eq. 19).

These particles are scattered in the intersection between two particle groups where a crack is desired, and an additional grid property c_i will be used for comparisons on the grid. If the crack criterion is violated for a particle, it will be invalidated and removed from the set P (and G).

4.4. Crack failure mode

Only grid nodes where at least two grids collide are considered for cracks. We determine if a node is cracked or not using a force based glue criteria with respect to the relative grid force $\Delta \boldsymbol{f}_{gi}^k$. For our purposes we will disregard forces due to compression and we remove the inwards force component with respect to the contact normal $\boldsymbol{n}_{(b,d)\boldsymbol{i}}^k$. This limits us to considering only tensile crack forces. The relative grid force is then

$$\begin{aligned}\Delta \boldsymbol{f}_{gi}^k = &(\boldsymbol{f}_{bi}^k - \min(0, \boldsymbol{f}_{bi}^k \cdot \boldsymbol{n}_{(b,d)\boldsymbol{i}}^k)\boldsymbol{n}_{(b,d)\boldsymbol{i}}^k) \\ &- (\boldsymbol{f}_{di}^k - \min(0, \boldsymbol{f}_{di}^k \cdot \boldsymbol{n}_{(d,b)\boldsymbol{i}}^k)\boldsymbol{n}_{(d,b)\boldsymbol{i}}^k).\end{aligned} \tag{20}$$

The state of the node is determined by comparing the size of $\Delta \boldsymbol{f}_g^k$ with the nodal threshold criterion $c_{\boldsymbol{i}}^k$, as acquired by rasterizing the glue particles to the grid:

$$c_{\boldsymbol{i}}^k = \frac{\sum_g c_g w_{g\boldsymbol{i}}^k}{\sum_g w_{g\boldsymbol{i}}^k}. \tag{21}$$

The introduction of the weight average in eq. 21 makes the threshold independent of the number of nearby glue particles. As a result it becomes easier to achieve the desired material break point. Lastly, we can determine if two grids should be merged at a grid node. If

$$|\Delta \boldsymbol{f}_{g\boldsymbol{i}}^k| < c_{\boldsymbol{i}}^k, \tag{22}$$

eq. 19 is used to merge grid node $\boldsymbol{i}$. Otherwise, the node is cracked and contact is automatically resolved by the multi-body solver. Additionally, if the crack criterion was violated, all glue particles that contributed to that grid node must be invalidated and removed from the set P. This information is backtracked from the recently cracked grid node using a particle index grid, which allows for fast grid-to-particle look-up, and the related glue particles are deleted from the simulation domain.

One can regard crack formation as composed of three linearly independent modes: an opening mode and two shear type modes. The present version of the crack algorithm only aims at proper simulation of cracks dominated by the opening mode. There are presently two issues that complicate the extension of the algorithm to properly simulate cracks where other modes dominate: (1) The crack failure criteria, eqs. 20, 21, are only accurate for tensile stress normal to the (pre-generated) crack. Such forces are only present for cracks of the opening-mode type, and it prevents cracks from forming when the other modes dominate. This could be resolved by, e.g., using a stress based criterion instead. However, stress calculations are normally done per particle before grid calculations take place, and no knowledge of contact normals exist yet. (2) However, with a crack failure criterion that allows shear-mode cracks to open, the resulting surfaces that now barely touch will trigger the impenetrable condition, eq. 1, giving artificial normal forces that push the crack open slightly. The situation is identical for internal cracks, which would leave a visual artefact similar to that seen in the fractured torus example in section 5.5. This problem of 0-width cracks is similar to, and discussed more thoroughly in, Mitchell et al. (2015) with level sets as the frame of reference.

4.5. The crack algorithm

In the present work we focus on simulation of bodies with a single crack. For the algorithm to allow branching cracks and bodies that break apart into more than two segments, in addition to what has been presented above, every glue particle must be assigned to exactly two bodies, and its crack failure critera, eqs. 20, 21, should be evaluated only with respect to those two bodies. While this seems a trivial extension of what has been presented so far, we have not implemented that extension.

The changes necessary to extend the proposed contact algorithm in section 3.2 for cracks are now few. See the steps below. Items with overlapping numbers are additions to the same step in the contact algorithm.

2. Additionally, glue particles need to be rasterized to a grid by eq. 21.

...

8. If the node is cracked as determined by eq. 22, delete glue particles that contributed to the grid node. Then, apply correction forces to grid pair.
9. Otherwise, merge grids at the node by eq. 19.

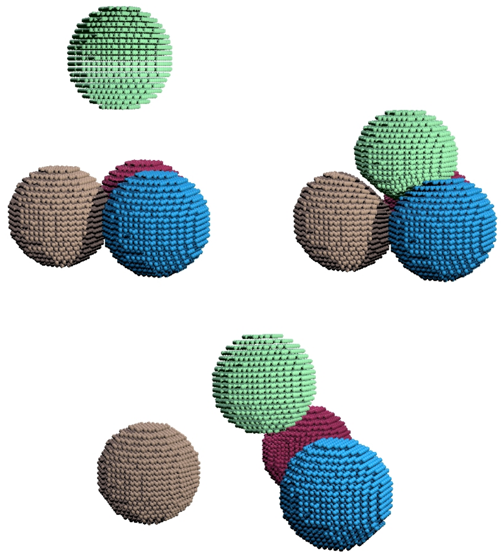

Fig. 4. A moving sphere colliding with three stationary spheres.

Sphere	y value [m]	$|\boldsymbol{v}|$ [m/s]
Blue	−9.04	1.46
Red	−8.93	1.45
Brown	−8.76	1.44

Table 1. ***Colliding sphere*** **data after 10 seconds of simulation.**

5. Results

We present the results of our method as follows: Sections 5.1, 5.2, and 5.3 relate to the contact algorithm. Our algorithm will perform in a manner identical to that of Huang et al. (2011) for two-body collisions, and we will solely focus on collisions with more than two bodies. Sections 5.4 and 5.5 show our algorithm for simulating cracks.

By using one separate grid for every body, $b_1, ..., b_n$, there will effectively need to be n sets of values at every grid node. Implementing this using a pre-allocated dense grid would render the method impractical for even modest sized simulations due to rapid increase in memory usage. We recommend using sparse grids, which dynamically allocate memory and typically reduce the overall memory footprint of the simulation. All the simulations were performed using the OpenVDB framework [Museth (2013)].

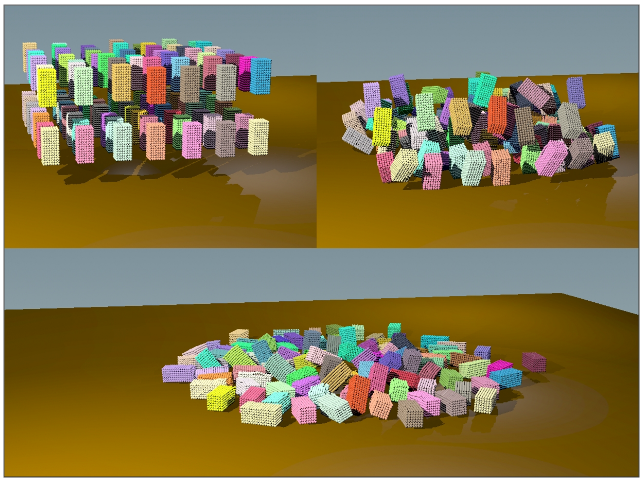
Fig. 5. 98 *Kubbs* dropped on the ground.

5.1. *Colliding spheres*

We first consider a moving sphere colliding with three stationary spheres positioned each with its center on a vertex of an equilateral triangle as in fig. 4. Even though the particles of each sphere are separated their grids connect. In the second picture (top right) there are grid nodes where the three grids are colliding. The triangle lies in the horizontal plane, defined as $y = 0$. The colliding sphere has been positioned in the middle of the triangle 3 meters above, and is advancing with a speed of 3 m/s. No gravity is applied. Each sphere has a radius of 1 m and has been discretized into 3 911 points each using an uniform grid spacing of 20 cm. The friction has been set to $\mu = 0.2$. After 10 seconds the y values of the initially non-moving spheres and their velocities can be seen in table 1.

5.2. *Kubb*

Two sets of slightly jittered bricks, kubbs[1], are stacked on top of each other separated by a small distance. See fig. 5. They are dropped from an average distance of 0.4 and 0.9 meters respectively, with gravity applied. The dimensions of one brick is $15 \times 30 \times 15\text{cm}^3$, and it is discretized by 735 points with a grid spacing of 4 cm. There are a total of 98 separate bodies (and 98 separate grids) and a total of 72 030 particles in the simulation. The cubic kernel from Stomakhin et al. (2013) rasterizes each particle to 4 grid nodes in each dimension, and the grid representation of each brick will extend around 8 cm further than the particles in each direction. As a result, it is not uncommon to find grid nodes where 5 or 6 bodies collide. The bricks are dropped from a standstill.

The ground has a small mound to create a more interesting simulation (barely visible in the top left image). After 10 seconds the bricks have come to a rest.

5.3. *Sand brush*

Our method is not restricted to merely one type of material. Consider fig. 6; here we have coupled an elastic brush together with plastic sand. These will interact dynamically in the simulation using our collision algorithm without any extra coupling. The sand has been modelled using the Drucker-Prager plasticity model as described by Klar et al. (2016). It consists of 230 913 particles, and they have been colored depending on their initial y-displacement to create a sand-like feeling.

The motion of the brush is determined by the shaft which has been modelled as a boundary. The boundary imposes the condition $\boldsymbol{v}_{bi}^k = \boldsymbol{v}_{ci}^k$ for all touched grid nodes, and the elastic material will loosely follow the boundary. The brush is discretized into 3 603 points with a Young's modulus of 1 MPa and Poisson's ratio of 0.1 to emulate a rubbery-like material.

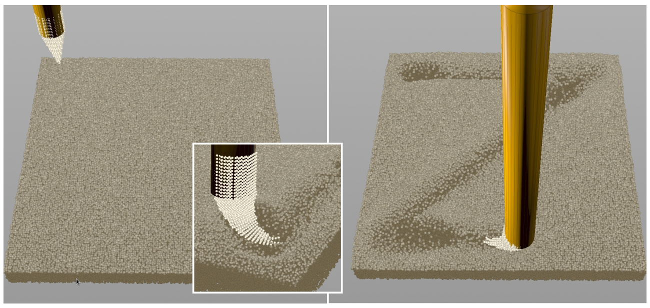
Fig. 6. An elastic brush moving through sand.

5.4. *Controlled tearing*

The first example aims to investigate if our method is successful in directing how a material cracks. Consider a rectangular brick as in fig. 7. It is discretized into 35 301 material points, and a small cut-out has been created on one side to create a material weakness that will help initiate the crack. The same simulation with identical material parameters was run three different times, with the only difference being how the material has been pre-fractured.

The left-most specimen has not been pre-fractured at all. As can be seen the material still breaks in the middle as expected, and it has smaller fractures close to the top plate due to high forces close to the boundary. The other two simulations have the crack interface modified by the pre-fracture process; the middle has a diagonal cut and the right-most has a crack in a zigzag pattern. Glue particles with a threshold of

[1]Kubb is a lawn game frequently played in Sweden. The game involves knocking over a set of wooden blocks, similar in shape to the bricks shown in the image.

$c_g \in [140, 160]$ N have been uniformly scattered over the crack surface.

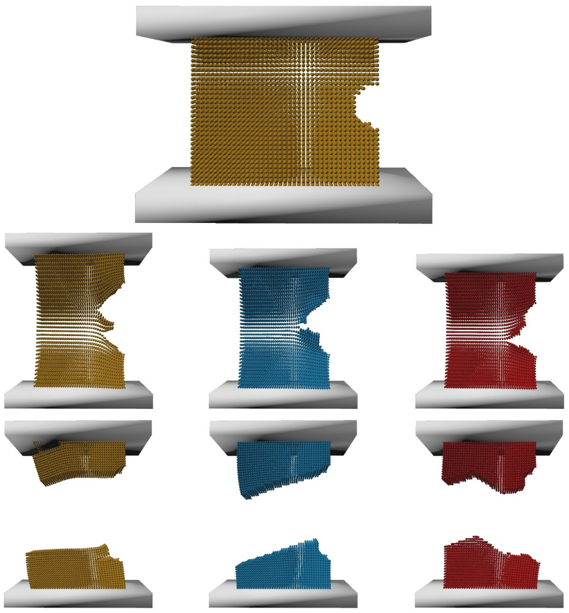
Fig. 7. Tearing of a sponge-like material. The left most setup have not been pre-fractured, and represents how normal MPM will behave. The blue and red has been pre-fractured using a diagonal and zigzag pattern respectively, thus demonstrating that this method can be used for controlled crack propagation.

5.5. *Internal cracks*

Consider a torus as in fig. 8. It has been pre-fractured into two pieces by a horizontal plane splitting it in the middle and consists of two symmetric halves; one as the bottom and one on top. Glue particles have been scattered on the outer half of both disks that represent the interface where the bottom and top parts meet. As a result, the inner halves of the two disks constitute two open internal cracks. The torus approaches the right wall with a velocity of 10 m/s and consists of 37 100 points. The outer radius is 50 cm and the internal is 25 cm. The simulation is performed without gravity.

6. Discussion

6.1. *Multi-body solver*

The colliding sphere example, fig. 4, shows largely a natural many-body collision. However, seemingly separate bodies are colliding, which is the result of our impenetrability condition, eq. 1. This is an artifact of the method—grid nodes should in reality be allowed to penetrate each other, but doing so effectively forces contact to be resolved on a particle level. This proves difficult, as previous simple operations such as determining contact and calculating normals instead become unfeasible. Furthermore, there is a notable discrepancy in vertical displacement due to different velocities, see table 1. This error is introduced with the assumption that all contact normals at a grid node are orthogonal, which in practice is rare, and could be resolved by solving the optimization problem eq. 18.

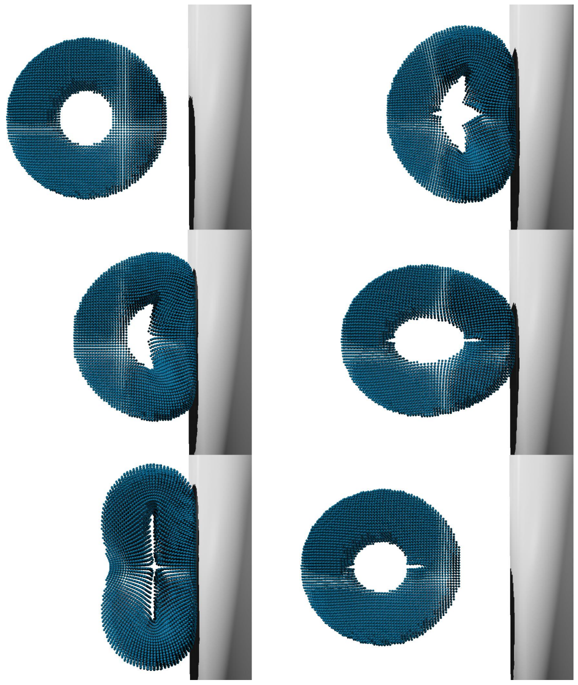
Fig. 8. A fractured torus colliding with a wall. The simulation time frames are ordered top to bottom, left to right.

Alternatively, another Eularian contact algorithm, like the one presented in Levin et al. (2011), could be employed instead.

Our collision algorithm is indifferent to the constitutive model and is a standalone addition to the MPM framework. As such it can be used in conjunction with any number of different materials, and it allows for an easy coupling of materials to create dynamic scenarios as can be seen by the *Sand brush* example. Combining more than one material type in the same simulation is not something specific to our collision resolution scheme, however, by introducing separate grids we can ensure that no blending of the materials will occur. As such, running the *Sand brush* simulation without our algorithm would result in sand particles (unnaturally) sticking to the brush.

6.2. *Cracks*

The focus of this development was to create realistic looking fracturing. Figure 7 shows that it is possible to direct the fracture process using our technique. Emphasis has been given especially to making the setup phase of pre-fracturing easy. Due to eq. 21 not much thought has to be given to the distribution of glue particles in the crack intersection; as long as all distances to nearby glue particles are less than the radius of the glue particles, the setup will function properly.

However, the torus example in fig. 8 demonstrates a shortcoming with our method. The problem of contact being determined prematurely by the multi-body solver was previously discussed. The same artefact is present also in the simulation of the torus, and is seen by noticing that a crack never fully closes after the initial rupture when the torus collides with the wall. Nairn (2003) addresses the problem of premature contact by detecting collisions with respect to volume change instead of relative velocity. For internal cracks, this would allow the two contact surfaces to approach closer before numerical contact.

It would be beneficial to move the crack test to a Lagrangian frame of reference. Particle stress is calculated as part of the force update, and the method would improve from using a stress based measure instead of the current force based one in eq. 20, as shearing and rotating forces are not treated properly. Additionally, this would simplify the removal of glue particles after they have been invalidated, as the particle criterion c_g could be used instead of the grid based one. The difficulty is in transferring the information of contact normals from the grid to the particles. CRAMP solves this issue by a line crossing algorithm, as that effectively gives contact information in the Lagrangian frame.

The removal of the line crossing algorithm is the main computational speedup that this work provides over CRAMP, together with the fact that no crack release rates are calculated for crack propagation, as done in e.g., Bardenhagen et al. (2011). For real materials, a propagating crack tip is normally followed by a plastic deformation in the crack region, that absorbs some of the potential energy released when the crack forms. The method presented here does not take this deformation into account, and it may cause erroneous fracturing as a result. In our experiments, the overhead generated by our crack and multi-body extensions appears to be negligible, i.e. below a couple of percent of the run time of a standard MPM (see e.g., Stomakhin et al. (2013)).

7. Conclusions

We extended the basic MPM solver with a contact algorithm that can handle, in theory, any number of colliding bodies. Our method will produce the same result for 2-body collisions as Huang et al. (2011). When increasing the number of colliding bodies, artefacts may be introduced due to the approximation that pairwise contact normals are orthogonal, but the end result nevertheless tends to look natural. We have leveraged the multi-body solver to model cracks, defined as the intersection between parts of a body that have been pre-fractured. To indicate where a material is cracked *glue particles* are introduced. These are scattered in the crack intersection and resemble the Lagrangian mesh as used in CRAMP, however, glue particles are unconnected and do not represent a crack, but they are used by the Eulerian grid to determine if a grid node should be treated as cracked or not. A tearing scenario was presented in section 4 to show the applicability of this procedure.

Acknowledgments

This work was conducted as an internship at DreamWorks Animation. We thank Mihai Alden for valuable input and for help with the implementation, and Gergely Klar for insightful discussions on MPM. We also thank the anonymous reviewers for their helpful comments and improvement suggestions.

References

Sulsky, D, Zhou, S, Schreyer, H. Application of a particle-in-cell method to solid mechanics. Computer Physics Communications 1995;87(1):236 – 252. URL: `http://www.sciencedirect.com/science/article/pii/0010465594001707`. doi:`http://dx.doi.org/10.1016/0010-4655(94)00170-7`; particle Simulation Methods.

Huang, P, Zhang, X, Ma, S, Huang, X. Contact algorithms for the material point method in impact and penetration simulation. International journal for numerical methods in engineering 2011;85:498–517.

Terzopoulos, D, Fleischer, K. Modeling inelastic deformation: Viscolelasticity, plasticity, fracture. SIGGRAPH Comput Graph 1988;22(4):269–278. URL: `http://doi.acm.org/10.1145/378456.378522`. doi:`10.1145/378456.378522`.

O'Brien, J, Hodgins, J. Graphical modeling and animation of brittle fracture. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '99; New York, NY, USA: ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-48560-5; 1999, p. 137–146. URL: `http://dx.doi.org/10.1145/311535.311550`. doi:`10.1145/311535.311550`.

O'Brien, J, Bargteil, A, Hodgins, J. Graphical modeling and animation of ductile fracture. ACM Trans Graph 2002;21(3):291–294. URL: `http://doi.acm.org/10.1145/566654.566579`. doi:`10.1145/566654.566579`.

Pauly, M, Keiser, R, Adams, B, Dutré, P, Gross, M, Guibas, L. Meshless animation of fracturing solids. ACM Trans Graph 2005;24(3):957–964. URL: `http://doi.acm.org/10.1145/1073204.1073296`. doi:`10.1145/1073204.1073296`.

Kaufmann, P, Martin, S, Botsch, M, Grinspun, E, Gross, M. Enrichment textures for detailed cutting of shells. ACM Trans Graph 2009;28(3):50:1–50:10. URL: `http://doi.acm.org/10.1145/1531326.1531356`. doi:`10.1145/1531326.1531356`.

Müller, M, Chentanez, N, Kim, T. Real time dynamic fracture with volumetric approximate convex decompositions. ACM Trans Graph 2013;32(4):115:1–115:10. URL: `http://doi.acm.org/10.1145/2461912.2461934`. doi:`10.1145/2461912.2461934`.

Stomakhin, A, Schroeder, C, Chai, L, Teran, J, Selle, A. A material point method for snow simulation. ACM Trans Graph 2013;32(4):102:1–102:10. URL: `http://doi.acm.org/10.1145/2461912.2461948`. doi:`10.1145/2461912.2461948`.

Ram, D, Gast, T, Jiang, C, Schroeder, C, Stomakhin, A, Teran, J, et al. A material point method for viscoelastic fluids, foams and sponges. 2015. URL: `http://doi.acm.org/10.1145/2786784.2786798`. doi:`10.1145/2786784.2786798`.

Yue, Y, Smith, B, Batty, C, Zheng, C, Grinspun, E. Continuum foam: A material point method for shear-dependent flows. ACM Trans Graph 2015;34(5):160:1–160:20. URL: `http://doi.acm.org/10.1145/2751541`. doi:`10.1145/2751541`.

Stomakhin, A, Schroeder, C, Jiang, C, Chai, L, Teran, J, Selle, A. Augmented mpm for phase-change and varied materials. ACM Trans Graph 2014;33(4):138:1–138:11. URL: `http://doi.acm.org/10.1145/2601097.2601176`. doi:`10.1145/2601097.2601176`.

Klar, G, Gast, T, Pradhana, A. Drucker-prager elastoplasticity for sand animation. ACM Trans Graph 2016;35:4.

Daviet, G, Bertails-Descoubes, F. A semi-implicit material point method for the continuum simulation of granular materials. ACM Trans Graph 2016;35(4):102:1–102:13. URL: `http://doi.acm.org/10.1145/2897824.2925877`. doi:`10.1145/2897824.2925877`.

Jiang, C, Schroeder, C, Selle, A, Teran, J, Stomakhin, A. The affine particle-in-cell method. ACM Trans Graph 2015;34(4):51:1–51:10. URL: `http://doi.acm.org/10.1145/2766996`. doi:`10.1145/2766996`.

Zhu, Y, Bridson, R. Animating sand as a fluid. ACM Trans Graph 2005;.

Levin, D, Litven, J, Jones, GL, Sueda, S, Pai, DK. Eulerian solid simulation with contact. ACM Trans Graph 2011;30(4):36:1–36:10. URL: `http://doi.acm.org/10.1145/2010324.1964931`. doi:`10.1145/2010324.1964931`.

Fan, Y, Litven, J, Levin, D, Pai, D. Eulerian-on-lagrangian simulation. ACM Trans Graph 2013;32(3):22:1–22:9. URL: `http://doi.acm.org/10.1145/2487228.2487230`. doi:`10.1145/2487228.2487230`.

Hegemann, J, Jiang, C, Schroeder, C, Teran, J. A level set method for ductile fracture. In: Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation. SCA '13; New York, NY, USA: ACM.

ISBN 978-1-4503-2132-7; 2013, p. 193–201. URL: `http://doi.acm.org/10.1145/2485895.2485908`. doi:`10.1145/2485895.2485908`.

Nairn, J. Material point method with explicit cracks. Tech Science Press: CMES 2003;4(6):640–663.

Guo, Y, Nairn, J. Three-dimensional dynamic fracture analysis using the material point method. Tech Science Press: CMES 2006;1:11–25.

Bardenhagen, S, Nairn, J, Lu, H. Simulation of dynamic fracture with the material point method using a mixed j-integral and cohesive law approach. International Journal of Fracture 2011;170:49–66.

Daphalapurkar, NP, Lu, H, Coker, D, Komanduri, R. Simulation of dynamic crack growth using the generalized interpolation material point (gimp) method. International Journal of Fracture 2007;143(1):79–102. URL: `https://doi.org/10.1007/s10704-007-9051-z`. doi:`10.1007/s10704-007-9051-z`.

Mitchell, N, Aanjaneya, M, Setaluri, R, Sifakis, E. Non-manifold level sets: A multivalued implicit surface representation with applications to self-collision processing. ACM Trans Graph 2015;34(6):247:1–247:9. URL: `http://doi.acm.org/10.1145/2816795.2818100`. doi:`10.1145/2816795.2818100`.

Museth, K. Vdb: High-resolution sparse volumes with dynamic topology. ACM Trans Graph 2013;32.