An Adaptive Generalized Interpolation Material Point Method for Simulating Elastoplastic Materials

MING GAO, University of Wisconsin Madison
ANDRE PRADHANA TAMPUBOLON, University of Pennsylvania
CHENFANFU JIANG, University of Pennsylvania
EFTYCHIOS SIFAKIS, University of Wisconsin Madison

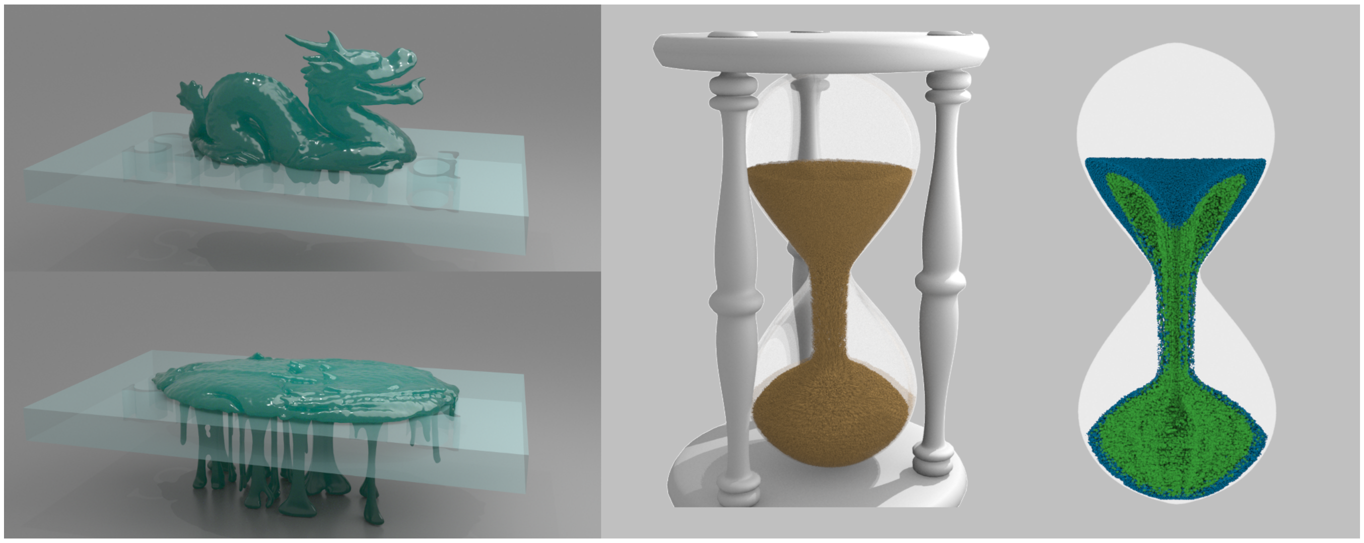

Fig. 1. Left: An elastoplastic model is dropped into a plane with a thin perforation pattern; our adaptive discretization allows the material to drip through. Right: Adaptive sand simulation with a visualization of the underlying grid refinement. We color refined particles with blue and coarse ones with green.

We present an adaptive Generalized Interpolation Material Point (GIMP) method for simulating elastoplastic materials. Our approach allows adaptive refining and coarsening of different regions of the material, leading to an efficient MPM solver that concentrates most of the computation resources in specific regions of interest. We propose a C^1 continuous adaptive basis function that satisfies the partition of unity property and remains non-negative throughout the computational domain. We develop a practical strategy for particle-grid transfers that leverages the recently introduced SPGrid data structure for storing sparse multi-layered grids. We demonstrate the robustness and efficiency of our method on the simulation of various elastic and plastic materials. We also compare key kernel components to uniform grid MPM solvers to highlight performance benefits of our method.

CCS Concepts: • **Computing methodologies → Physical simulation**;

Additional Key Words and Phrases: Material Point Method (MPM), Generalized Interpolation Material Point (GIMP), Adaptive grids, Elastoplasticity

ACM Reference format:
Ming Gao, Andre Pradhana Tampubolon, Chenfanfu Jiang, and Eftychios Sifakis. 2017. An Adaptive Generalized Interpolation Material Point Method for Simulating Elastoplastic Materials. *ACM Trans. Graph.* 36, 6, Article 223 (November 2017), 12 pages.
DOI: 10.1145/3130800.3130879

1 INTRODUCTION

The Material Point Method (MPM) has been attracting considerable interest since it was introduced to the field of computer graphics by Stomakhin et al. [2013]. Combining advantages from both Lagrangian particle representation and Eulerian grid representation, MPM proves to be especially effective for animating elastoplastic materials undergoing large deformation or topology change [Jiang et al. 2016]. Despite its physical realism and geometrical convenience, a traditional MPM solver has several disadvantages. First, it is more computationally expensive than mesh-based Lagrangian approaches such as those based on Finite Element Methods (FEM) [Sifakis and Barbic 2012]. The bottleneck of MPM is usually the costly transfer operations between the particles and the grid. The cost of such transfer operations is particularly evident when we realize that MPM has to maintain the same grid resolution and a sufficient particle count throughout the simulation domain. The overhead of this process is highlighted in scenarios such as the example of drawing in a

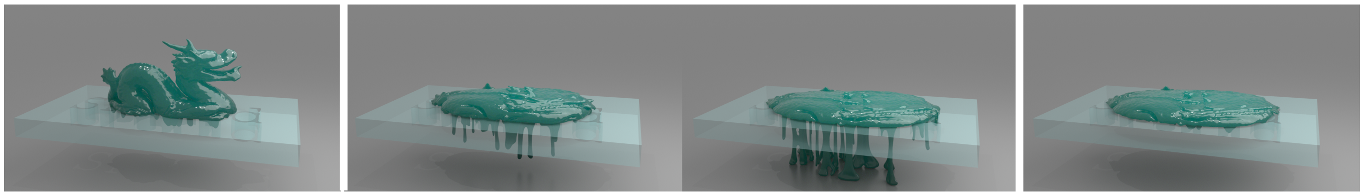

Fig. 2. **Dragon goo**. An elastoplastic dragon model is placed on a table carved with intricate thin slits. Our adaptive simulation (center) refines the background grid in the vicinity of the collision object, allowing the dragon to seep through, while a simulation on a uniform grid (right) cannot resolve the contact.

sandbox from Klár et al. [2016], where the majority of sand grains do not move at all.

Another disadvantage of traditional MPM is related to its ability to resolve (self-) collision events. MPM automatically enforces non-slip contact. However it treats two particles as being in contact whenever they affect some common grid nodes. As a result, the separation distance is inherently proportional to the grid cell spacing Δx. High resolution is required in order to prevent visually noticeable collision gaps even for the simulation with very simple dynamics. Furthermore, MPM cannot resolve a boundary condition that is finer than the grid resolution. This implies that materials cannot pass through holes smaller than Δx. Similarly, a blade that is thinner than Δx can not cut through materials.

We propose an adaptive variant of the Generalized Interpolation Material Point (GIMP) method [Bardenhagen and Kober 2004] to alleviate these limitations. By only refining regions of particular interest, we can resolve fine-grained self contact and object collision features with significantly reduced computational cost. We demonstrate the efficacy of our method on the simulation of various elastic and plastic materials. We summarize our main contributions as:

- The introduction of the GIMP paradigm to MPM simulation for computer graphics applications.
- An *adaptive* GIMP discretization framework with significantly improved shape functions to enforce important properties such as C^1 continuity and non-negativity.
- A memory efficient and highly regular parallel particle-grid transfer scheme that achieves attractive performance on both uniform and arbitrary *non-graded* adaptive grids. We show how all the necessary operations in our adaptive grids can be naturally paired with the SPGrid data structure, and implemented using efficient, uniform grid operations as building blocks.
- A highly optimized (threaded and vectorized) particle-grid transfer approach with a number of aggressive vectorization optimizations that were specifically enabled by our proposed perspective on implementing grid adaptivity via the multi-level SPGrid representation.

Our method is competitive with a uniform gird on a per-cell or per-particle basis. We also note that as opposed to the approach proposed by Lian et al.[2015] that assumes a *graded* adaptive Cartesian grid where neighboring cells do not differ by more than one refinement level, our approach in constructing an adaptive grid is relatively straightforward and simple to implement. Our grid adaptivity also remains efficient regardless of the complexity of refinement levels. Additionally, our transfer operators achieve a very rigorous standard of optimality, matching or exceeding the bandwidth of state-of-the-art parallel uniform MPM codes with the same grid cell and/or particle count.

Scope. The principal objective of this work is to propose an adaptive MPM scheme, equipping the weights with all desired properties, which also has the potential to realize competitive performance, and can facilitate and catalyze follow-up work on this topic. We admit, however, that our current framework does not prove that adaptive MPM will always deliver performance superior to uniform MPM, due to the fact that explicit time integration is arguably the least favorable for reaching aggressive end-to-end performance advantages over uniform schemes.

2 RELATED WORK

Adaptive simulation of deformable solids. The use of spatial adaptivity to mitigate the cost of simulating detailed elastic and elastoplastic bodies has been widely documented in the computer graphics literature. Debunne et al. [1999] proposed one of the early continuum-based adaptive volumetric simulation techniques using finite differences, departing from prior schemes based on mass-spring systems. Similar ideas were soon combined with Finite Element (FEM) Methods [Capell et al. 2002] where a hierarchical deformation basis would be constructed via subdivision, with individual bases activated or disabled based on deformation. Similarly, Grinspun et al. [2002] cater to adaptive FEM simulation of volumetric solids and shells by refining the deformation basis, rather than the elements themselves, providing a natural handling of T-junctions at resolution transitions. Topology change, instigated by cutting operations and material fracture provided a compelling context for adaptive simulation, with a number of authors exploring octree-based discretizations of elastic solids [Seiler et al. 2011] which also allowed the use of efficient multi-resolution solvers [Dick et al. 2011], while others combined octrees with shape matching techniques in modeling cutting operations [Steinemann et al. 2008]. Localized adaptation in response to high deformation provided one of the most natural motivations for Discontinuous Galerkin methods [Kaufmann et al. 2009].

Although several of the aforementioned techniques are based on octrees, a number of researchers focused on adaptive tetrahedral discretizations readily produced by robust meshing algorithms [Labelle and Shewchuk 2007] which have been very popular in modeling elastoplastic deformation [Wicke et al. 2010]. Specifically, adaptive tetrahedralizations based on BCC lattices have been used to model viscoelastic [Wojtan and Turk 2008] as well as hyperelastic materials [Sifakis et al. 2007]. Although contact and collision handling is often designed independently of adaptive refinement, in certain instances

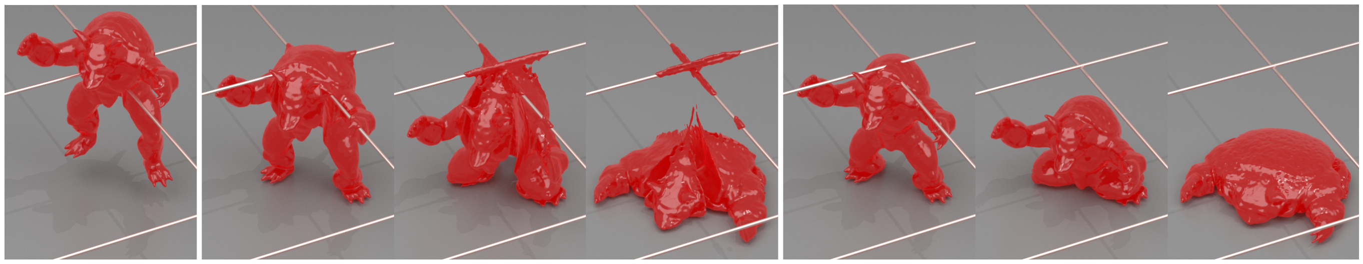

Fig. 3. **Armadillo wire cut**. A gooey armadillo is dropped through two thin intersecting wires. Left: the model just prior to collision; Center: grid refined to 4× the base resolution in the vicinity of the wires, per our method; Right: A uniform grid with comparable particle count largely misses the collision event.

as in the work of Otaduy et al. [2007] the two behaviors are tightly integrated as to concurrently accelerate elastic simulation and collision detection. Finally, for a deeper survey of adaptive techniques for deformable models, we refer to the excellent recent report by Manteaux et al. [2016].

Adaptive fluids. Adaptive fluid simulation has been studied extensively by previous approaches since the work of Losasso et al. [2004] on octree-based water and smoke. Tetrahedral based adaptive fluid simulation was combined with embedded boundary methods by Batty et al. [2010]. Ando et al. [2012] adopted particle splitting and merging for liquid sheets. Ando et al. [2013] simulated highly detailed splashes on a novel FLIP scheme discretized on an adaptive tetrahedral mesh. Ferstl et al. [2016] proposed a narrow band FLIP that couples with an Eulerian solver. Adaptive sampling methods were also developed for SPH [Adams et al. 2007; Solenthaler and Gross 2011].

Sparse and adaptive grid structures. Although pointer-based tree structures [Losasso et al. 2004] are the most straightforward choice for storing adaptive grids, the under-utilization of memory bandwidth associated with indirect access and suboptimal prefetching that is intrinsic to pointer-based trees has led researchers to explore alternative storage structures. RLE-based techniques [Chentanez and Müller 2011; Houston et al. 2006; Irving et al. 2006] combine the regularity of a 2D uniform grid with the compression of a 1D run-length encoding. Adaptive Mesh Refinement (AMR) techniques and variants thereof [Cohen et al. 2010; English et al. 2013; Patel et al. 2005] combine adaptivity and regularity by patching together uniform grids of different resolutions. One of the most efficient and broadly used adaptive data structures, OpenVDB [Museth 2013], is based on a tree with a high branching factor that yields large uniform grids at leaf nodes. Our present work is based on the recently developed Sparse Paged Grid (SPGrid) data structure [Setaluri et al. 2014], which targets in-core processing and exploits the hardware accelerated mechanisms that support the Virtual Memory subsystem of modern CPUs. Apart from a sparse grid structure, Setaluri et al. [2014] paired SPGrid with a different perspective of octrees, treating them as stacks of sparsely populated *uniform* grids, across which the cells of a geometric octree are scattered. This concept has been subsequently applied to simulation of high-resolution fluids [Liu et al. 2016] and hybridized with second-order accurate techniques for incompressible flow [Aanjaneya et al. 2017].

Material Point Methods. MPM [Sulsky et al. 1995] is a generalization of the hybrid Fluid Implicit Particle (FLIP) method [Brackbill et al. 1988; Bridson 2008; Zhu and Bridson 2005] to solid mechanics. It has proven to be a promising discretization choice for animating many solid materials including snow [Stomakhin et al. 2013], foam [Ram et al. 2015; Yue et al. 2015], sand [Daviet and Bertails-Descoubes 2016; Klár et al. 2016], cloth [Jiang et al. 2017] and solid-fluid mixture [Stomakhin et al. 2014]. The original GIMP concept [Bardenhagen and Kober 2004] was described in the context of uniform grids. There is some engineering literature exploring adaptive MPM. Early work of Tan et al. [2002] mainly focused on particle splitting. Some initial ventures into coupling GIMP and adaptivity [Daphalapurkar et al. 2007; Ma et al. 2006, 2005] managed to enforce both partition of unity and C^1 continuity, but their applications were restricted to nested grids (not a general octree). Lian et al. [2014] employed the concept of embedding fine T-junctions in coarse parents, but only along the surfaces separating resolution levels. More recently, Lian et al. [2015] developed the mesh-grading MPM (MGMPM) that modified the shape functions while maintaining the partition of unity. However their method, as we will see later, is prone to generating negative interpolation weights in certain scenarios, risking potential instability. The shape functions in both [Lian et al. 2014] and [Lian et al. 2015] are not based on GIMP and only C^0 continuous. Our method, on the other hand, guarantees all desired properties: partition of unity, non-negativity and C^1 continuity for the weights.

3 GENERALIZED INTERPOLATION MATERIAL POINT METHOD

The Generalized Interpolation Material Point (GIMP) method, proposed by Bardenhagen and Kober [2004], is a generalization of the original MPM to allow a wider range of interpolation functions between the particles and the grid.

3.1 Governing equations

Before discussing GIMP, we first briefly review the original MPM. We refer to [Stomakhin et al. 2013] for more details. The simulated material is perceived as a continuum body which is a subset of $\mathbb{R}^d$. Here d denotes the spatial dimension, which can be 2 or 3. At any give time t, there is a deformation mapping $\boldsymbol{\varphi}(\cdot, t) : \mathbb{R}^d \rightarrow \mathbb{R}^d$, which maps points in the undeformed configuration to a deformed configuration. More precisely, a point $\mathbf{X} \in \Omega^0$ is mapped to $\mathbf{x}(\mathbf{X}, t) =$

$\boldsymbol{\varphi}(\mathbf{X}, t) \in \Omega^t$. The deformation gradient $\mathbf{F} = \partial \boldsymbol{\varphi} / \partial \mathbf{X}$ describes the material deformation and acts as a common strain measure [Bonet and Wood 2008]. We denote the determinant of the deformation gradient by $J := \det(\mathbf{F})$.

The governing equations we need to solve are the conservation of mass and conservation of momentum. They are written as

$$\frac{D\rho}{Dt} + \rho \nabla \cdot \mathbf{v} = 0 \text{ and } \rho \frac{D\mathbf{v}}{Dt} = \nabla \cdot \boldsymbol{\sigma} + \rho \mathbf{g} \tag{1}$$

respectively, where D/Dt is the material derivative ($\frac{Df}{Dt} := \frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla f$ for a generic scalar function f), ρ is the density, $\mathbf{v}$ is the velocity, $\mathbf{g}$ is gravity, $\boldsymbol{\sigma}$ is the Cauchy stress. We assume that there exists an elastic energy density function $\Psi(\mathbf{F})$, so that the Cauchy stress can be written as $\boldsymbol{\sigma} = \frac{1}{J} \frac{\partial \Psi}{\partial \mathbf{F}} \mathbf{F}^T$ (see Section 4 for more discussion on the physical model).

3.2 Discretization

First, we explain the notation used in this paper. Subscript i is an index that enumerates grid nodes, while p is an index that enumerates discrete particles. Certain quantities relating grid nodes and particles will carry a double subscript (ip), as will be the case with interpolation functions and weights. Superscript n identifies the discrete time step associated with a time-varying quantity.

In MPM, particles carry attributes such as mass (m_p), position ($\mathbf{x}_p$), velocity ($\mathbf{v}_p$), deformation gradient ($\mathbf{F}_p$), and other material parameters. A background grid is used as a scratch pad to discretize and solve the governing equations. The communication between particles and grid information is handled through an interpolation function. We denote the weight and the corresponding gradient between particle p and grid node i with w_{ip} (a scalar) and ∇w_{ip} (a vector), respectively.

Here we provide an overview of MPM stages in a time step, assuming an explicit symplectic-Euler time integration. GIMP follows exactly the same procedure.

(1) **Particle to grid.** Assuming we are at time n, particle mass and momentum are transferred from particles to grid nodes with $m_i^n = \sum_p m_p w_{ip}^n$ and $(m\mathbf{v})_i^n = \sum_p m_p \mathbf{v}_p^n w_{ip}^n$. The velocity of node i is computed as $\mathbf{v}_i^n = (m\mathbf{v})_i^n / m_i^n$ when $m_i^n \neq 0$. It is set to $\mathbf{0}$ otherwise.
(2) **Compute grid forces.** This comes from discretizing the conservation of momentum. For node i, the force is given by $\mathbf{f}_i^n = m_i^n \mathbf{g} - \sum_p V_p^0 J_p^n \boldsymbol{\sigma}_p^n \nabla w_{ip}^n$, where V_p^0 is the undeformed particle volume.
(3) **Grid velocity update.** Denoting the updated grid node velocities with $\hat{\mathbf{v}}_i$, symplectic Euler computes it as $\hat{\mathbf{v}}_i = \mathbf{v}_i^n + \Delta t \mathbf{f}_i^n / m_i^n$.
(4) **Collision treatment.** $\hat{\mathbf{v}}_i$ for each grid node is further processed for nodes inside collision objects. The relative velocity is set to $\mathbf{0}$ inside "sticky" collision objects. The normal component is set to $\mathbf{0}$ in "slip" collision objects.
(5) **Strain evolution.** The particle deformation gradient $\mathbf{F}_p$ evolves as $\mathbf{F}_p^{n+1} = \left(\mathbf{I} + \Delta t \sum_i \hat{\mathbf{v}}_i (\nabla w_{ip}^n)^T\right) \mathbf{F}_p^n$.
(6) **Grid to particle.** Particle velocities and positions are updated from grid velocities with $\mathbf{v}_p^{n+1} = \alpha(\mathbf{v}_p^n + \sum_i (\hat{\mathbf{v}}_i - \mathbf{v}_i^n) w_{ip}^n) + (1 - \alpha) \sum_i \hat{\mathbf{v}}_i w_{ip}^n$ and $\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \Delta t \sum_i \hat{\mathbf{v}}_i w_{ip}^n$, where $\alpha = 0.95$ [Stomakhin et al. 2013].

3.3 From MPM to GIMP

Traditional MPM [Sulsky et al. 1995] defines the weight w_{ip} between particle p and grid node i to be exactly the interpolation function $N_i(\mathbf{x})$ of node i evaluated at $\mathbf{x}_p$:

$$w_{ip} = N_i(\mathbf{x}_p). \tag{2}$$

Accordingly the weight gradient is $\nabla w_{ip} = \nabla^{\mathbf{x}} N_i(\mathbf{x}_p)$. Here $N_i(\mathbf{x})$ is the standard finite element trilinear basis function in 3D (bilinear in 2D) defined for node i.

While this choice of particle-grid weights works fine for problems with small deformation, Steffen et al. [2008] observes that it suffers from the "cell crossing instability" for practical problems. This is because piecewise linear basis functions are only C^0 continuous at cell boundaries. The corresponding gradient ∇w_{ip} is therefore discontinuous. Two apparent problems are associated with this property. First of all, it is possible that a huge force is exerted on a node with tiny mass, causing numerical issues. Secondly, discontinuity of the gradient implies discontinuity of the force (cf. Step 2 in the the MPM/GIMP stages). Noise and instability may thus occur as a particle travels across cells. These reasons render this choice of basis function to be ineffective for traditional MPM. We note that this is not a problem for other hybrid particle-grid methods such as a FLIP fluid solver because the weight gradient is not needed.

One convenient solution to remedy the cell crossing instability is to use higher order C^1 or even C^2 continuous interpolation functions such as quadratic or cubic B-splines as in Stomakhin et al. [2013]. Unfortunately there are not many other choices of the interpolation function, given the following constraints:

- **Partition of unity:** $\sum_i w_{ip} = 1, \forall \mathbf{x}_p$. This is required for mass and momentum conservation [Jiang et al. 2015].
- **Non-negativity:** $w_{ip} \geq 0$.
- **Interpolation:** $\mathbf{x}_p = \sum_i w_{ip} \mathbf{x}_i$.
- C^1 **continuity:** w_{ip} needs to be at least C^1 continuous.
- **Local support:** w_{ip} is only non-zero for $\mathbf{x}_p$ near $\mathbf{x}_i$.

The last requirement is primarily due to practical considerations. The non-negativity constraint rules out the possibility of high order FEM basis functions such as 9-node quadratic quadrilateral elements or 8-node serendipity quadrilateral elements in 2D (and their corresponding shape functions for hexahedral meshes in 3D) [Hughes 2012]. As we will discuss more in detail in Section 5.1, allowing negative weights may cause serious issues when mass on a grid node becomes negative [Andersen and Andersen 2007].

GIMP is another alternative to eliminating the cell crossing instability. Instead of choosing $N_i(\mathbf{x})$ to be C^1, GIMP constructs weights with C^1 continuity, using as building blocks bases $N_i(\mathbf{x})$ with just C^0 continuity, often chosen to be the standard multilinear basis.

The full simulation domain is denoted by Ω. Associated to a particle p is the notion of particle domain Ω_p surrounding it (as yet of unspecified shape; will be chosen to be an axis aligned box in our implementation). Thus, the volume of a particle can be computed as

$$\hat{V}_p = \int_{\Omega \cap \Omega_p} d\mathbf{x}. \tag{3}$$

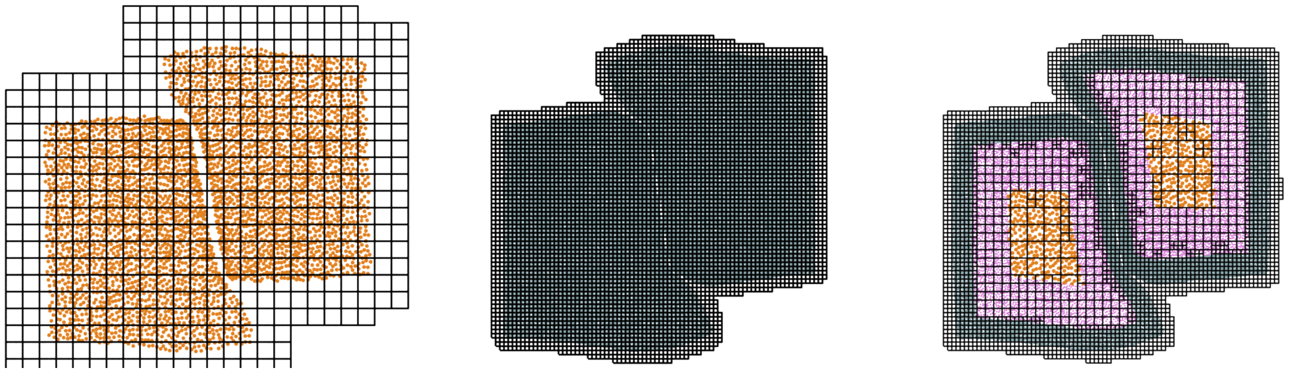

Fig. 4. **Colliding 2D jello squares**. Two hyperelastic jello squares are driven to collide with each other. Left: A uniformly coarse grid leaves large separation gaps. Middle: A 4×-refined uniform grid nicely resolves the contact, at the expense of gratuitous computation in the interior. Right: Our adaptive scheme.

Unlike Equation 2, GIMP defines the weight to be

$$w_{ip} = \frac{1}{\hat{V}_p} \int_\Omega \chi_p(\mathbf{x}) N_i(\mathbf{x})\, d\mathbf{x}, \tag{4}$$

with $\chi_p(\mathbf{x})$ being the particle characteristic function defined over the particle domain Ω_p.

The traditional MPM is a special case of GIMP where $\chi_p(\mathbf{x}) = \hat{V}_p \delta(\mathbf{x} - \mathbf{x}_p)$. Common GIMP schemes often choose $\chi_p(\mathbf{x})$ to be the characteristic indicator function centered at $\mathbf{x}_p$, i.e.,

$$\chi_p(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \Omega_p, \\ 0 & \text{otherwise.} \end{cases}$$

We model Ω_p as an axis-aligned box centered at $\mathbf{x}_p$. Another common simplification that is often made is that Ω_p does not rotate or deform. This is also known as uGIMP (undeformed GIMP). Thus the integral is always evaluated in a box with size $\hat{V}_p = L_p^d$, where L_p is the fixed side length of particle p's box and $d = 2$ or 3 is the dimension. Under this choice of $\chi_p(\mathbf{x})$, Equation 4 simplifies to

$$w_{ip} = \frac{1}{\hat{V}_p} \int_{\Omega_p} N_i(\mathbf{x}) d\mathbf{x}. \tag{5}$$

Similarly, we define

$$\nabla w_{ip} = \frac{1}{\hat{V}_p} \int_{\Omega_p} \nabla^{\mathbf{x}} N_i(\mathbf{x}) d\mathbf{x}. \tag{6}$$

Recall that $N_i(\mathbf{x})$ is piecewise linear and C^0 continuous. Since the weights result from a convolution of $N_i(\mathbf{x})$ with the indicator function of Ω_p, w_{ip} becomes C^1 continuous and ∇w_{ip} is C^0 continuous if they are viewed as functions of $\mathbf{x}$. Additionally, w_{ip} satisfies all constraints mentioned previously.

GIMP weights are inherently smooth. We note that GIMP further recovers traditional MPM with quadratic B-spline interpolation on a uniform grid when $L_p = \Delta x$. We can see this from the convolution definition of uniform B-spline functions. More specifically, assuming $\Delta x = 1$ in 1D, the B-spline of order-0 is given by

$$N^0(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in [0, 1], \\ 0 & \text{otherwise.} \end{cases}$$

If we convolute $N^0(\mathbf{x})$ with itself as $N^1(\mathbf{x}) = N^0(\mathbf{x}) * N^0(\mathbf{x})$, we can see $N^1(\mathbf{x})$ is exactly the piecewise linear order-1 B-spline with support $[-1, 1]$. We can further write $N^2(\mathbf{x}) = N^1(\mathbf{x}) * N^0(\mathbf{x})$ and $N^3(\mathbf{x}) = N^2(\mathbf{x}) * N^0(\mathbf{x})$, or generally $N^k(\mathbf{x}) = N^{(k-1)}(\mathbf{x}) * N^0(\mathbf{x})$ to recursively construct uniform higher order B-splines. Consequently, by noticing $N^0(\mathbf{x}) = \chi_p(\mathbf{x})$ when $L_p = \Delta x$, we see that the GIMP shape function and the quadratic B-spline shape function are identical in this case. This result generalizes to 2D and 3D due to the property that multi-dimension B-splines are simply tensor products of univariate 1D B-splines.

Even though B-splines MPM and GIMP can be made equivalent on a uniform grid, it is much more difficult to generalize high order B-splines to an adaptive grid. Much of the difficulty lies in the treatment of T-junctions and transition grid cells between different levels. On the other hand, the GIMP view from Equation 4 is naturally defined on an adaptive grid, as long as a C^0 continuous piecewise linear function $N_i(\mathbf{x})$ is well defined. We show how to robustly construct such basis functions in Section 5.

4 ELASTOPLASTICITY

We adopt finite strain elastoplasticity to model different material behaviors. In this section we follow [Bonet and Wood 2008] and assumes that the deformation gradient $\mathbf{F}$ is decomposed into elastic and plastic parts as $\mathbf{F} = \mathbf{F}^E\mathbf{F}^P$. In this paper we only consider (i) purely elastic objects, (ii) elastoplastic von Mises materials, and (iii) granular Drucker-Prager sand. However, our approach can be easily combined with any constitutive models.

4.1 Hyperelasticity

In hyperelasticity setting, there exists an elastic energy density function $\Psi(\mathbf{F}^E)$ that penalizes the deviation of $\mathbf{F}^E$ from a pure rotation. As such, the first Piola-Kirchhoff stress is determined by $\mathbf{P} = \frac{\partial \Psi}{\partial \mathbf{F}^E}(\mathbf{F}^P)^{-T}$. The Cauchy stress can be obtained as $\boldsymbol{\sigma} = \frac{1}{J}\mathbf{P}\mathbf{F}^T$. In this paper, we decided to choose the fixed-corotated model [Stomakhin et al. 2012] for purely elastic solids due to its robustness under large deformation, where Ψ is defined as $\Psi(\mathbf{F}^E) = \mu\|\mathbf{F}^E - \mathbf{R}^E\|_F^2 + \frac{\lambda}{2}(J^E - 1)^2$, where $\mathbf{R}^E$ is the rotation tensor from the polar decomposition $\mathbf{F}^E = \mathbf{R}^E\mathbf{S}^E$, $J^E = \det(\mathbf{F}^E)$, μ and λ are Lamé parameters.

For elastoplastic materials, it is often more convenient to adopt the Hencky strain $\boldsymbol{\epsilon} = \frac{1}{2}\log(\mathbf{F}\mathbf{F}^T)$ from Mast et al.[2013] and to use the St. Venant-Kirchhoff energy density

$$\Psi(\mathbf{F}^E) = \mu \mathrm{tr}(\log(\Sigma^E)^2) + \frac{1}{2}\lambda(\mathrm{tr}(\log(\Sigma^E)))^2, \tag{7}$$

where Σ comes from SVD of $\mathbf{F}^E$: $\mathbf{F}^E = \mathbf{U}^E\Sigma^E(\mathbf{V}^E)^T$.

4.2 Plasticity

In this paper we consider two plasticity models, namely the Drucker-Prager model with a non-associative flow rule for granular materials, and the von Mises model with an associative flow rule for perfectly plastic materials. We use Equation 7 for the elastic response.

4.2.1 Drucker-Prager. The Drucker-Prager yield surface can be derived by applying the Mohr-Coulomb friction law in a continuum. The admissible stress is inside the region given by $y(\boldsymbol{\sigma}) = c_F \mathrm{tr}(\boldsymbol{\sigma}) + \|\boldsymbol{\sigma} - \frac{\mathrm{tr}(\boldsymbol{\sigma})}{d}\mathbf{I}\|_F \leq 0$, where c_F is the coefficient of internal friction. Discretely, if the trial stress is outside the yield region, it is projected back onto the yield surface through an closed-form solution of the return mapping [Klár et al. 2016]. The original return mapping causes non-physical volume gain when the material is under expansion. We adopt the volume correction treatment as in Tampubolon et al. [2017] to eliminate this artifact.

4.2.2 von Mises. Von Mises plasticity is widely used for metal and ductile materials. It is also closely related to computer graphics plasticity models that were successfully applied for simulating plastic flow and gooey materials [Bargteil et al. 2007; Wojtan and Turk 2008]. In 3D, the von Mises yield surface is defined as $y(\boldsymbol{\sigma}) = \sqrt{3J_2} - \sigma_y \leq 0$, where σ_y is the yield stress, $J_2 = \frac{1}{2}\mathbf{s} : \mathbf{s}$ is the second invariant of the deviatoric stress $\mathbf{s}$. Here $\mathbf{s} := \boldsymbol{\sigma}^{dev} = \boldsymbol{\sigma} + p\mathbf{I}$ and $p = -\frac{1}{3}\mathrm{tr}(\boldsymbol{\sigma})$. Unlike the Drucker-Prager yield surface for granular materials, von Mises plasticity uses an associative flow rule. The discrete flow rule thus chooses $\frac{\partial y}{\partial \boldsymbol{\sigma}}$ as its direction. In the principal stress space, the shape of von Mises yield surface is an infinitely long cylinder centered at the hydrostatic axis ($\boldsymbol{\sigma}_1 = \boldsymbol{\sigma}_2 = \boldsymbol{\sigma}_3$, where $\boldsymbol{\sigma}_i$ is the i'th eigenvalue of $\boldsymbol{\sigma}$). Following the same methodology in [Klár et al. 2016], we derive a closed form solution of the return mapping for von Mises plasticity. Assuming $\boldsymbol{\epsilon}^E := \log(|\mathbf{F}^E|)$ is the trial Hencky strain in the principal space, when the trial stress is outside the yield region, we project $\boldsymbol{\epsilon}^E$ to the yield surface to obtain a new Hencky strain $\mathbf{H}^E = \boldsymbol{\epsilon}^E - \delta\gamma \frac{\mathrm{dev}(\boldsymbol{\epsilon}^E)}{\|\mathrm{dev}(\boldsymbol{\epsilon}^E)\|}$, where $\delta\gamma = \|\mathrm{dev}(\boldsymbol{\epsilon}^E)\| - \frac{\sigma_y}{2\mu}$ and $\mathrm{dev}(\boldsymbol{\epsilon}^E) = \boldsymbol{\epsilon}^E - \frac{\mathrm{tr}(\boldsymbol{\epsilon}^E)}{3}\mathbf{I}$.

5 ADAPTIVE BASIS FUNCTIONS

We employ a spatially adaptive computational grid. As the simulation proceeds, the grid must locally refine and coarsen to hierarchical refinement levels. Instead of trying to define a C^1 continuous B-spline-like interpolation function that covers all cases and T-junctions, we degrade the problem into defining a C^0 function. As discussed in Section 3.3, GIMP will use convolution to convert the C^0 continuous interpolation function $N_i(\mathbf{x})$ to a C^1 continuous shape function w_{ip}.

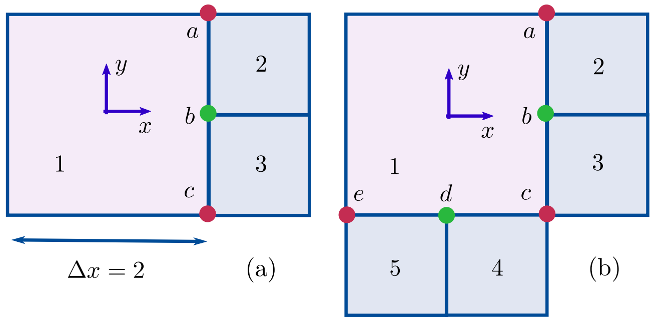

Fig. 5. **Illustration of free T-junction.** A test case for the adaptive grid basis with free T-junctions described in Section 5.1. The interpolation functions remain non-negative in the case of (a). However, in the (b) case where there exist two T-junction nodes, N_c may turn negative and become problematic.

5.1 Adaptive grid basis with free T-junctions

We start by describing the state-of-the-art multi-level grid approach by Lian et al. [2015]. Cell spacing of the grid of level n is $\Delta x_n = \Delta x_0/2^n$, where Δx_0 is the coarsest cell spacing. This results in additional hanging nodes at T-junctions at the interface between cells of different refinement levels.

Taking the case of Figure 5(a) as an example, cell 2 and 3 are of a level higher than cell 1. Node b is a T-junction node at the level transition interface. In the formulation of Lian et al. [2015] it is treated as an actual degree of freedom on the grid. Inside cell 2 and 3, the interpolation functions associated with nodes a, b, c are simply the standard bilinear hat functions. For example, if we denote the hat function basis of node γ in cell k with H_γ^k, and the grid basis interpolation function with N_γ^k, we have

$$N_a^2 = H_a^2,\ N_b^2 = H_b^2,\ N_b^3 = H_b^3,\ N_c^3 = H_c^3. \tag{8}$$

In other words, within cells that have no T-junctions in their periphery, the shape functions of their corner vertices are the standard bilinear hat functions; this is not the case, however for cells containing T-junctions (such as cell 1 in Figure 5), neither for the corner vertices or T-junction nodes associated with such cells. N_b^1 must be defined in a way that ensures continuity of N_b along edge $a - b - c$. If we parameterize cell 1 with a coordinate system spanning $[-1, 1]^2$ with the origin at its center and assume node b is at $(x = 1, y = 0)$, then $N_b^1 = \frac{1}{2}(1 + x)(1 - |y|)$. To enforce continuity of N_a, N_b and partition of unity, the interpolation function of node a and b inside cell 1 are constructed as $N_a^1 = H_a^1 - \frac{1}{2}N_b^1$, $N_c^1 = H_c^1 - \frac{1}{2}N_b^1$. A similar strategy can be adopted in 3D, where a transition cell may have 8 to 26 nodes, depending on the number of T-junctions.

While this approach works fine for the case of Figure 5(a), it starts to break the non-negativity constraint in the case of Figure 5(b) where there exists another T-junction node d. Using the same local coordinate system defined above, we have

$$N_b^1 = \frac{1}{2}(1 + x)(1 - |y|), \qquad N_d^1 = \frac{1}{2}(1 - |x|)(1 + y), \tag{9}$$

$$H_c^1 = \frac{1}{4}(1 + x)(1 - y), \qquad N_c^1 = H_c^1 - \frac{1}{2}N_b^1 - \frac{1}{2}N_d^1. \tag{10}$$

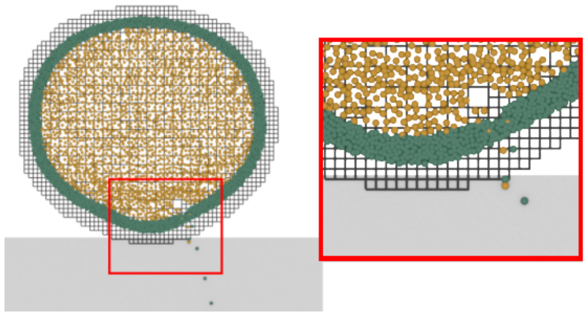
Fig. 6. Negative weights causing instability in a sphere dropping experiment.

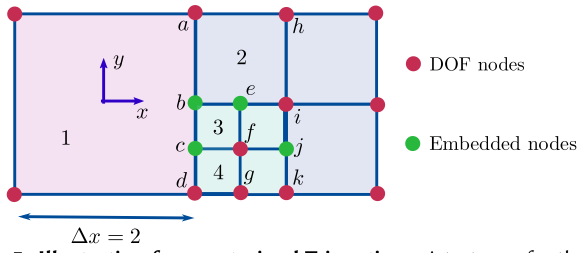

Fig. 7. **Illustration for constrained T-junctions.** A test case for the our adaptive grid basis with constrained T-junctions, as described in Section 5.2.

It is easy to see that N_c^1 can sometimes become negative, for example when $x = 0.5$ and $y = 0.1$. As a result, some grid nodes may have negative mass after the particles-to-grid transfer. Negative weights in MPM causes instability and severe loss of accuracy [Andersen and Andersen 2007]. For example, we could construct a case with two particles affecting a grid node with exactly opposite weights. As a result, the nodal mass becomes zero, resulting in an incorrect value of grid node velocity. Subsequently, the grid node velocity update and grid-to-particles transfer will be erroneous, causing the simulation to go unstable or behave non-physically. Figure 6 shows a practical case where such instability happens at a region where negative nodal mass exists.

5.2 Adaptive grid basis with constrained T-junctions

To prevent negative interpolation weights, we propose a different strategy for defining a C^0 continuous $N_i(\mathbf{x})$ for GIMP. The key idea is the same with the classical constrained hanging node treatment in octree FEM simulations [Fries et al. 2011; Legrain et al. 2011]. In contrast to the approach of Lian et al. [2015], our treatment does not construct interpolation functions on the T-junctions. T-junction nodes are constrained to move with their parent nodes at cell corners. Therefore, T-junction nodes are embedded vertices that do not belong to the set of real degrees of freedom.

We shall use the grid topology shown in Figure 7 to illustrate our approach; note that this arrangement includes a number of real degrees of freedom (colored red) as well as T-junctions (colored green). Initially, we will ignore any constraints that the T-junction nodes might be subjected to, and associate a shape function with all enumerated vertices, real or T-junction. For a given node γ we define a shape function $H_\gamma = \sum_n H_\gamma^n$ as the summation of all the

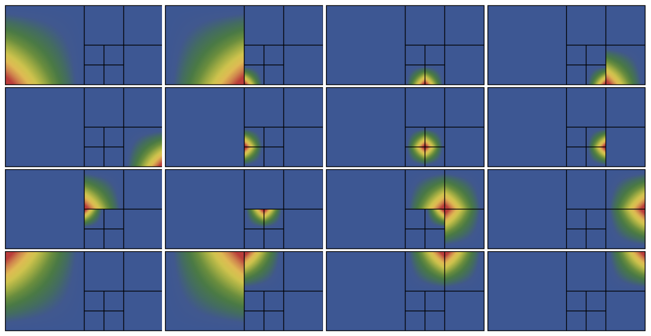
Fig. 8. Illustration of the shape functions H_γ defined in Section 5.2.

standard bilinear hat functions H_γ^n, summed over all cells (indexed by n) for which γ is a *corner vertex* (not a T-junction). For example, we would thus have $H_a = H_a^1 + H_a^2$ and $H_b = H_b^2 + H_b^3$ in the scenario of Figure 7. The resulting functions H_γ are plotted in Figure 8. Using this basis (in the absence of any constraints), a scalar field $q(\mathbf{x})$ would be interpolated from nodal values as $q(\mathbf{x}) = \sum_\gamma q_\gamma H_\gamma(\mathbf{x})$. Of course, the reconstructed function $q(\mathbf{x})$ would be discontinuous (since the shape functions H_γ are discontinuous themselves, most notably along cell faces that contain T-junctions). Restoring continuity would require us to enforce constraints on the nodal values q_γ, such as $q_b = \frac{1}{2}q_a + \frac{1}{2}q_d$, $q_c = \frac{1}{4}q_a + \frac{3}{4}q_d$, $q_e = \frac{1}{4}q_a + \frac{1}{4}q_d + \frac{1}{2}q_i$ and $q_j = \frac{1}{2}q_i + \frac{1}{2}q_k$ in our specific example. Substituting these constraints into equation $q(\mathbf{x}) = \sum_\gamma q_\gamma H_\gamma(\mathbf{x})$ replaces this summation with a different expression $q(\mathbf{x}) = \sum_\eta q_\eta N_\eta(\mathbf{x})$, where now η enumerates only the *real* (non T-junction) degrees of freedom, and the new functions $N_\eta(\mathbf{x})$ have been formed by accumulating partial contributions from the constrained degrees of freedom that were eliminated after the substitution. For our specific example of Figure 7, we would have

$$N_a = H_a + \frac{1}{2}H_b + \frac{1}{4}H_c + \frac{1}{4}H_e, \tag{11}$$

$$N_d = H_d + \frac{3}{4}H_c + \frac{1}{2}H_b + \frac{1}{4}H_e, \tag{12}$$

$$N_i = H_i + \frac{1}{2}H_e + \frac{1}{2}H_j, \tag{13}$$

$$N_k = H_k + \frac{1}{2}H_j, \tag{14}$$

while the remaining non-T-junction nodes satisfy $N_\eta = H_\eta$. Note that the contribution from a constrained degree of freedom to a real degree of freedom (such as from H_c to N_a) directly results from the corresponding nodal value constraint (such as q_c). We use these newly defined functions $N_\eta(\mathbf{x})$ as our shape basis; these functions are fully C^0 continuous, as illustrated in Figure 9. We show a more detailed explanation of how one computes the shape for the basis functions and prove their properties (continuity and partition of unity) in the supplementary technical document [Gao et al. 2017].

We highlight the dual perspective : we can envision this construction either as an alteration of the basis functions of real (non T-junction) nodes, or as the result of constraining the coefficients of the shape functions H_γ associated with all nodes, to enforce embedding constraints; this duality will be exploited in the next section.

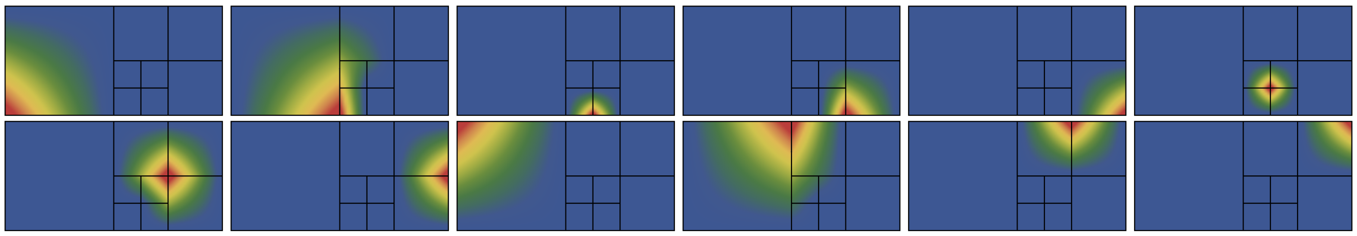

Fig. 9. Visualization of our C^0 continuous shape function basis, defined in Section 5.2. Shape functions are only associated with non-T-junction nodes.

Each modified interpolation function is a linear combination of hat basis functions. When we construct the GIMP shape functions and their gradients through Equation 4 and 6, the resulting w_{ip} is C^1 continuous, and ∇w_{ip} is C^0 continuous. In practice, instead of explicitly constructing the modified interpolation functions, we use a ghost cell based multi-layer scattering approach that is efficient and only require simple uniform grid operations (see Section 6.1).

6 ACCELERATED GRID-PARTICLE TRANSFERS

6.1 Multi-layer embedding and interpolation

In the previous section, we momentarily treated T-junctions as actual degrees of freedom, and endowed them with associated shape functions (H_γ). We will see that it is beneficial to take this exercise one step further. Consider the quadtree topology depicted in Figure 10. For the purposes of this hypothetical exercise, imagine that we fully subdivide the coarse cell on the left (with vertices $x^{2h}_{00}, \ldots, x^{2h}_{11}$) into four finer cells, straddling vertices now referred to as $x^h_{00}, \ldots, x^h_{22}$ (we are free to consider collocated vertices i.e. x^{2h}_{10} and x^h_{20} either as aliases of one another, or duplicates). Imagine that,

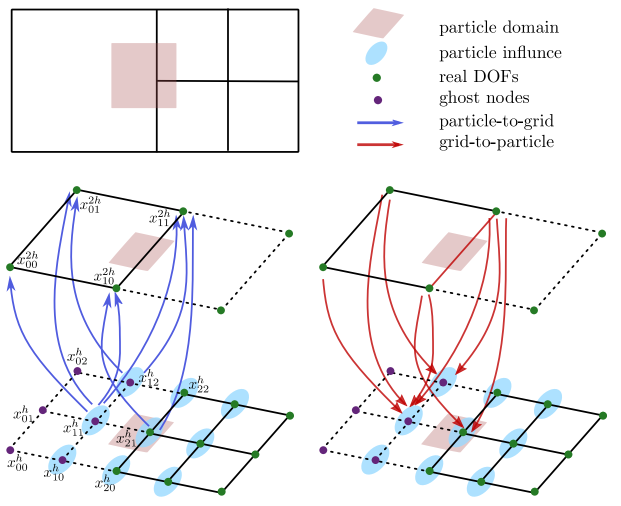

Fig. 10. Particle-grid transfer operations mapped to a multi-level sparse grid structure; operations involving applications of weights are performed solely on a uniform grid, and information is propagated across resolutions via the embedding relation. The approach trivially extends to non-graded grids.

instead of defining the starting shape functions H_γ on just the nodes that appear on the octree, we defined them on *all refined nodes* (i.e. define a function H^h_{ij} for every refined node x^h_{ij}). Refined nodes obey similar embedding relationships as previously seen for T-junctions, for example $x^h_{11} = \frac{1}{4}(x^{2h}_{00} + x^{2h}_{10} + x^{2h}_{01} + x^{2h}_{11})$. As in the previous section, we can construct the basis functions of the real degrees of freedom, by eliminating all refined nodes that are not collocated with a real node in the quadtree; this would yield for example

$$N^{2h}_{11} = H^h_{22} + \frac{1}{2}H^h_{12} + \frac{1}{2}H^h_{21} + \frac{1}{4}H^h_{11}.$$

This linear relation could be leveraged to define multi-level weights via the GIMP convolution, resulting for example in an application of Equation 4 to an expression of the form:

$$w^{2h}_{11,p} = w^h_{22,p} + \frac{1}{2}w^h_{12,p} + \frac{1}{2}w^h_{21,p} + \frac{1}{4}w^h_{11,p} \tag{15}$$

Consider the repercussions of this construction: All weights of the form $w^h_{ij,p}$ exhibit high regularity, as they result from the integration of the standard bilinear hat function on a *uniform* grid; this is aggressively leveraged in the vectorization optimization discussed in the following subsection. Finally, consider what Equation 15 implies for particle-to-grid transfers such as the mass update $m_i = \sum_p m_p w_{ip}$. By substituting Equation 15 in this expression, we see that the mass contributions could be equivalently computed in a two-step process: we first compute partial contributions $m^{2h}_{ij} = \sum_p m_p w^{2h}_{ij,p}$, and then distribute these contributions to the real quadtree degrees of freedom by multiplying them with the respective embedding weights, as shown in the left part of Figure 10.

We facilitate this hierarchical transfer by introducing the concept of ghost nodes in Figure 10, which are similar in motivation with those formulated by Setaluri et al. [Setaluri et al. 2014] to facilitate Laplacian stencil application, but different in the number of them that needs to be instanced. In the vicinity of any T-junction, we create duplicate refined variables by subdividing every cell incident to the T-junction down to the finest level touched by the T-junction. Among such nodes, those that are collocated with a real (non-constrained) quadtree node are labeled as "real" degrees of freedom; the remaining ones are labeled "ghost" and they serve as conduits for implementing the transfer operations in a hierarchical fashion. For any particle p, we detect the finest level intersecting the particle domain Ω_p, and carry out the particle-to-grid transfer exclusively to the refined nodes, some of which will be "ghost". After all particles have transferred their values to the appropriate uniform level, ghost node values are distributed in bulk to their coarser parents (multiplied by their embedding weights), until this process reaches all real quadtree nodes. The opposite process is followed for the grid-to-particle transfer; grid values of ghost nodes are interpolated from their embedding "real" parents, and then the weighted transfer to the particle is evaluated exclusively at the finest level intersecting Ω_p, as shown in the bottom-right part of Figure 10.

6.2 Vectorized weight computation

Our hierarchical approach to grid-particle transfers provides two significant regularity properties that can be leveraged to accelerate the computation of transfer weights, which can form the bulk of the computational burden if implemented inefficiently, as they need to be updated at every time step. First, since the actual transfer (modulo the embedding-based distributions) always takes place at a uniform grid, there is always a full set of 8 grid cells/27 grid nodes (in 3D) that participate in this transfer. All eight of such cells reside at the same level of resolution, and their contribution to the weight stencils can be computed in parallel, via SIMD instructions.

In addition to the aforementioned property, our code exploits yet another opportunity for SIMD computation: Since the shape functions in all cells involved in our transfers have now been reduced to the standard trilinear basis, we can consider the possibility of computing the weights of all eight *corner nodes* of every cell in parallel, using SIMD instructions. We illustrate how this materializes for the weights themselves, although the process easily extends to the weight gradients as well. Absorbing the characteristic indicator function into the integral of Equation 4 results in

$$w_{ijk,p} = \frac{1}{\hat{V}_p}\int_{\Omega\cap\Omega_p} N_{ijk}(\mathbf{x})\,d\mathbf{x} = \frac{1}{\hat{V}_p}\int_a^b\int_c^d\int_e^f N_{ijk}(\mathbf{x})\,dxdydz$$

where we have explicitly used a triple index $(i, j, k) \in \{0, 1\}^3$ to refer to the eight vertices of a cell, and $\Omega \cap \Omega_p = [a, b] \times [c, d] \times [e, f]$. Since the integrand is separable, it can be easily computed in closed form, e.g. for $N_{111} = \frac{1}{dx^3}xyz$ we have

$$W(a,b,c,d,e,f) := \frac{1}{\hat{V}_p}\int_{\Omega\cap\omega_p} N_{111}(\mathbf{x})d\mathbf{x} = \frac{(b^2-a^2)(d^2-c^2)(f^2-e^2)}{8\hat{V}_p dx^3}$$

We can easily observe that the integral of the shape functions N_{ijk} other than the one given here, can be easily computed by performing a change of variable $x_i \leftarrow 1 - x_i$ for one or more of the coordinate indices $i = 1, 2, 3$. This yields the following results:

$$\begin{aligned}
w_{000,p} &= W(1-b, 1-a, 1-d, 1-c, 1-f, 1-e)\\
w_{001,p} &= W(a, b, 1-d, 1-c, 1-f, 1-e)\\
w_{010,p} &= W(1-b, 1-a, c, d, 1-f, 1-e)\\
w_{011,p} &= W(a, b, c, d, 1-f, 1-e)\\
w_{100,p} &= W(1-b, 1-a, 1-d, 1-c, e, f), \text{ etc...}
\end{aligned}$$

We use this property, in conjunction with the regularity of computation across the eight cells involved, to implement a SIMD-optimized weight computation that computes the weights of all 8 cell vertices at once, by executing a vectorized implementation of the expression $W(\cdot)$ for a properly adjusted set of integration bounds.

The feasibility of these SIMD optimization, and the ability to structure the entire transfer using uniform operations as a building block, was a direct consequence of how our theoretical interpolation scheme was designed. The SIMD optimization would also directly benefit traditional GIMP on a uniform grid.

7 GRID RASTERIZATION AND PARTICLE RESAMPLING

7.1 Grid rasterization

We index grid levels with $1 \le q_g \le Q_g$ and particle types with $1 \le q_p \le Q_p$, where lower case denote finer grid resolution/smaller particles. In pre-process, particles attributes (e.g. position, mass, volume, and type) are prescribed by the user. Our convention is to place coarser grid resolution (hence sparser particle distribution) deep inside, and finer grid resolution (hence denser particle distribution) closer to the free surface or collision boundary. Static grid adaptivity (e.g. Figure 2 and 3) is accomplished in a pre-process step. However, dynamic grid adaptivity (e.g. Figure 4, 12, and 13) needs to be computed for each time step.

For hyperelasticity, the grid and particle adaptations are completely determined by the particle types. Starting from the coarsest level, the smallest particle type q_p within a cell of the current grid q_g is computed. We refine the cell if $q_p < q_g$ and the number of particles of each sub-cell is no less than particles per cell prescribed.

For elastoplastic material, we need to perform particle resampling. We first compute the approximate Manhattan distance d from the "free surface" for each finest cell. Then starting from the finest grid level, we merge sub-cells into a single cell only if they are beyond a prescribed distance criterion. However, this simple strategy is prone to the mismatch of particle type and grid level which can easily lead to numerical fracture (Lian et al. [2015]). Another problem is that for highly energetic motion (c.f. top right corner of Figure 11(a)), sparse particles in the interior are easily driven to the surface, which diminishes surface details.

7.2 Particle resampling

To alleviate these problems, we modify the split-and-merge approach proposed by Yue et al. [2015] to better suit our framework. Split-and-merge is applied to a particle depending on both its particle type and the corresponding Manhattan distances of the cell containing the particle. We define three distance parameters d_{small}, d_{medium} and d_{large}, if $d < d_{\text{small}}$, neither split nor merge of particles is enforced, otherwise the operations will be visually noticeable. Split is necessary when $d_{\text{small}} < d < d_{\text{medium}}$ to retain a detailed surface while merge is required for $d > d_{\text{large}}$ to reduce computational cost. In the last case, $d_{\text{medium}} < d < d_{\text{large}}$, split-and-merge depends on the particular application. Moreover, split-and-merge is prohibited if the number of particles within a cell is either too large or too small which could end up with the sparse particles moving closer to the surface (cf. blue circle of Figure 11(b)). For applications with more than two levels, (e.g. Figure 2 and 3), we repeat the process between each level of transition.

Split. To split a particle of type q to four particles of type $q - 1$ in 2D (eight in 3D), we first put a square(or a cube in 3D) centered at the original particle position with half diagonal length $\frac{dx}{4}$ (dx is the size of the cell containing the particle), then randomly rotate the square/cube and place the new particles in the vertices of the square/cube. To preserve both mass and momentum, the mass and

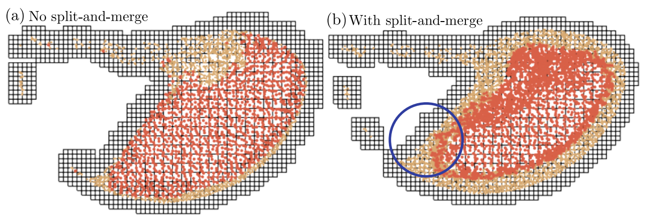

Fig. 11. **Sand in a rotating circle.** Top: particle split-and-merge turned off. Bottom: particle split-and-merge turned on.

	P2G 1 Core	G2P 1 Core	P2G 4 Cores	G2P 4 Cores
Dense	3.58	6.84	1.38	1.85
OpenVDB	2.39	5.8	0.64	1.54
SPGrid	3.05	1.26	0.82	0.28

Table 1. **Benchmark.** We compare our accelerated particle-to-grid (P2G) and grid-to-particle (G2P) transfers to a dense uniform grid MPM solver [Klár et al. 2016] and a sparse OpenVDB-based MPM solver [Tampubolon et al. 2017] on an Intel(R) Core(TM) i7-4770R.

volume are equally distributed to all new particles while the velocity and the deformation gradient are directly duplicated.

Merge. To merge particles, we start with a single particle and then search for its closest neighbors of the same particle type. The position of the new particle is the geometric center of the fine ones. The mass and volume are accumulated while the velocity is computed from a mass-weighted average. For deformation gradient, we first do the singular value decomposition $\mathbf{F}^E = \mathbf{U}^E \Sigma^E (\mathbf{V}^E)^T$. The matrix $\hat{\Sigma}$ is computed from the average of the Σ^E's. We apply quaternion average to the $\mathbf{U}^E$'s and $\mathbf{V}^E$'s to get $\hat{\mathbf{U}}^E$ and $\hat{\mathbf{V}}^E$. Finally, we compute the new deformation gradient as $\hat{\mathbf{F}}^E = \hat{\mathbf{U}}^E \hat{\Sigma}^E (\hat{\mathbf{V}}^E)^T$.

8 RESULTS

We list the performance and simulation parameters of our 3D simulations in Table 2, in which we also compare against uniform grid simulations. The first two examples (Jello, Figure 4 and Hourglass, Figure 13) illustrate the speedup of the adaptive approach against uniform methods that yield comparable visual detail with e.g., [Klár et al. 2016]. We highlight that the somewhat modest speedup is due to the fact that we gratuitously refine around the entirety of the free surface, rather than localizing refinement on regions undergoing contact. The last two examples (Dragon, Figure 2 and Armadillo, Figure 3) show that, when spending comparable computation effort with uniform techniques (with similar particle counts), our adaptive simulation can resolve significantly higher detail. Note that uniform MPM cannot handle our finest Δx case within reasonable amount of time and memory usage. Figure 12 shows colliding adaptive jellos in 3D. For this test, the memory usage is $0.18GB$ for the adaptive case and $0.24GB$ for the uniform dense case. The cost of particle-grid transfer operations in relate to the cost of a whole time step is 50% and 70% for them respectively. Figure 14 illustrates the fact that our adaptive simulation (left) produces a visually similar detailed dynamics of colliding jellos compared to a regular-dense-grid simulation (right).

Benchmark. We compare our accelerated particle-grid transfers to reliable implementations of the MPM solvers in [Klár et al. 2016] (with a dense uniform grid data structure) and [Tampubolon et al. 2017] (with the OpenVDB [Museth 2013] sparse grid) on an Intel(R) Core(TM) i7-4770R. We create a benchmark example with $\Delta x = \frac{1}{128}$. The particles are uniformed sampled on a grid from $(\frac{1}{8}, \frac{1}{8}, \frac{1}{8})$ to $(\frac{7}{8}, \frac{7}{8}, \frac{7}{8})$ with spacing $\frac{1}{256}$, for a total of just over 7 million particles. SPGrid (http://www.cs.wisc.edu/~sifakis/project_pages/SPGrid.html) was sourced from the project website, while the other MPM benchmarks were graciously shared with us by the respective authors. Table 1 lists the timing comparison. We additionally release our accelerated transfer code in the supplementary materials.

Contact. The benefit of using an adaptive discretization include better resolution of the contact between different MPM objects (as well as self contact). In Figure 4, we show a comparison of 2D colliding jellos on a uniform coarse grid, a uniform dense grid, and our adaptive grid. Adaptivity allows us to achieve very small separation distance. The adaptive discretization reduces the necessary computational cost compared to the uniform dense case.

Dynamic Adaptation. We simulate dry sand with the Drucker-Prager plasticity model and the free surface based adaptation criterion (Figure 1 and Figure 13). Both the particles and the grid are refined near the free surface and coarsened in the interior region. The computational resources are thus focused on the visible part, enabling highly detailed flow resolution.

Small Features. We further demonstrate the effectiveness of our method in resolving small scale collision object features. In Figure 2 we carve thin cracks on a glass table and put a dragon-shaped von Mises goo on it. We choose a small grid resolution for efficiency. A uniform grid cannot fully resolve the thin feature. Our method successfully let the goo to slip through the cracks by simply refining the cells near the crack. The robustness of our method on non-graded adaptivity allows us to refine the same cell for multiple times without needing to specially take care of neighboring cells. Note that the refinement is only performed to an extremely small portion of the computational domain and does not cause much overhead. Figure 3 shows another example where we cut an armadillo with two wires. Similarly, our method resolves the thin wire with a slight increase in computational cost. Although we adopt three successive levels of refinement in both of these two examples, all cells are either at the very coarsest or very finest level of resolution with a non-graded transition between them.

9 DISCUSSION

Limitations. Even though our adaptive scheme promises significant performance and detail benefits, it also carries a number of intrinsic limitations. Taking full advantage of adaptivity in simulations involving intricate contact would often require dynamically refining parts of the surface involved in collision events at any point in time (as opposed to preemptively refining the entire surface). In those cases, visual artifacts due to resampling might be evident (especially for simulation of granular materials, e.g. sand) and would require special attention to mitigate. The GIMP convolution allowed C^0 multilinear bases to be boosted to C_1 continuity in the computed

	levels (adaptive)	Δx (finest;adaptive)	particle # (adaptive)	time/step (adaptive)	Δx (uniform)	particle # (uniform)	time/step (uniform)	ρ	E	ν	σ_y	friction angle
Jello (Fig. 12)	3	1/256	8.8×10^5	1.67	1/256	1.74×10^6	2.74	1000	8e3	0.3	-	-
Hourglass (Fig. 13)	2	1/256	8.1×10^5	1.14	1/256	1.68×10^6	2.84	1000	1e5	0.3	-	30
Dragon (Fig. 2)	3	1/512	1.567×10^6	2.78	1/150	1.67×10^6	2.83	800	8e3	0.3	10	-
Armadillo (Fig. 3)	3	1/512	5.2×10^5	1.15	1/128	5.2×10^5	0.85	800	8e3	0.3	10	-

Table 2. **Simulation performance and parameters.** (ρ is material density, E is Young's modulus, ν is Poisson's ratio and σ_y is plasticity yield stress.) (1) Jello and Hourglass are run with Intel(R) Xeon(R) CPU E5-2687W v3. We compare the finest level Δx, first frame particle count, and per-step cost of our adaptive scheme with the simulation on a uniform grid that has the same resolution with our finest grid level. (2) Dragon and Armadillo are run with Intel(R) Xeon(R) CPU E5-1650 v3. Sampling on a uniform grid using our finest grid Δx causes a total number of over 30 million particles and cannot be handled within reasonable amount of time. Therefore we adjust the uniform grid resolution to match the particle count of our adaptive simulation, while maintaining the same particle-per-cell count.

weights; however, seeking a higher order of continuity is highly nontrivial in the adaptive case, as it would likely require a basis that is C^1 pre-convolution. Our use of a hierarchy of sparse grids allowed us to contain costs associated with expansive uniform grids; nevertheless, tracking the set of particles as they transition across cells of different resolution requires geometric search structures (e.g. box hierarchies) whose traversal and update can be less parallel-friendly than what would be necessary for the uniform case.

Scope restrictions. Our initial exploration of adaptive GIMP was consciously restricted in scope to simple simulations of elastoplasticity and refinement rules. In particular, refinement was triggered by simplified heuristics, such as the proximity to collision objects and/or the free surface. We have not investigated more intricate refinement criteria such as those triggered by large values of strain, or spatially localized to regions undergoing self-collision, for which an accurate and efficient detection would be less trivial. We have also restricted our investigation to purely explicit time integration techniques, and did not consider scenarios of fracturing elastic bodies, for which the use of adaptation would be naturally motivated.

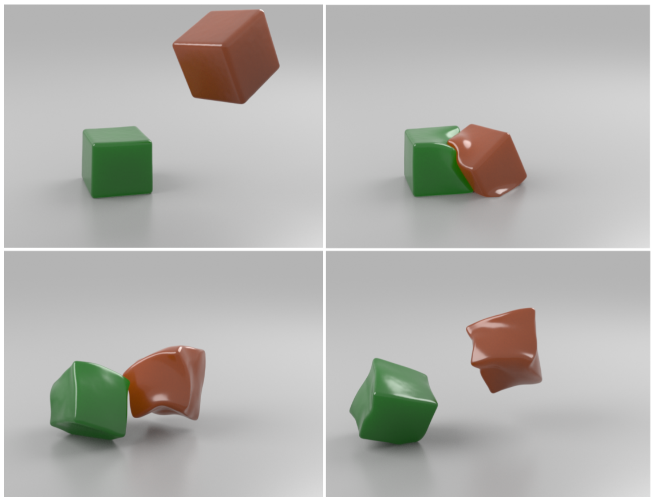

Fig. 12. **3D colliding jellos.** Two colliding jellos are simulated, where we placed finer particles and grid resolution near the free surface of the material.

Future work. We look forward to investigating extensions to implicit MPM approaches and evaluate if the SPGrid paradigm can deliver similar accelerations to operator evaluations in assembly-free iterative solvers. Extensions to MPM fluids and coupling behaviors between solid/granular/fluid phases would also be an exciting direction of investigation. We would also like to investigate the incorporation of the Affine Particle-In-Cell (APIC) method [Jiang et al. 2015] to our adaptive scheme for improved stability and angular momentum conservation. Finally, investigating combined use of adaptivity with fracture scenarios and detailed self-collision would be a very interesting (and likely nontrivial) future thread.

ACKNOWLEDGMENTS

We would like to thank Prof. Xiong Zhang for useful discussions. E.S. and M.G were supported by NSF grants IIS-1253598 and CCF-1533885. UPenn authors gratefully acknowledge the GPU donation from NVIDIA Corporation and a gift from Awowd, Inc.

REFERENCES

M. Aanjaneya, M. Gao, H. Liu, C. Batty, and E. Sifakis. 2017. Power Diagrams and Sparse Paged Grids for High Resolution Adaptive Liquids. *ACM Trans Graph.* 36, 4 (July 2017).

B. Adams, M. Pauly, R. Keiser, and L. Guibas. 2007. Adaptively sampled particle fluids. In *ACM Trans Graph*, Vol. 26. ACM, 48.

S. M. Andersen and L. Andersen. 2007. Material-Point Method Analysis of Bending in Elastic Beams. In *Inter Conf Civil, Struct Env Eng Comp.*

R. Ando, N. Thurey, and R. Tsuruno. 2012. Preserving fluid sheets with adaptively sampled anisotropic particles. *IEEE Trans Vis Comp Graph* 18, 8 (2012), 1202–1214.

R. Ando, N. Thurey, and C. Wojtan. 2013. Highly adaptive liquid simulations on tetrahedral meshes. *ACM Trans Graph* 32, 4 (2013), 103:1–103:10.

S. G. Bardenhagen and E. M. Kober. 2004. The generalized interpolation material point method. *Comp Mod in Eng and Sci* 5, 6 (2004), 477–496.

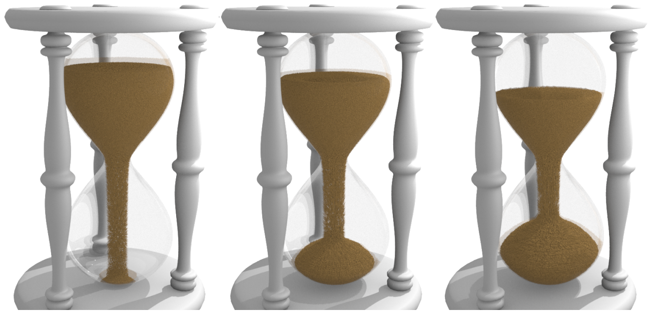

Fig. 13. **Sand in an hourglass.** Sand in an hourglass is simulated with finer particles and grid resolution strategically placed near the collision boundary.

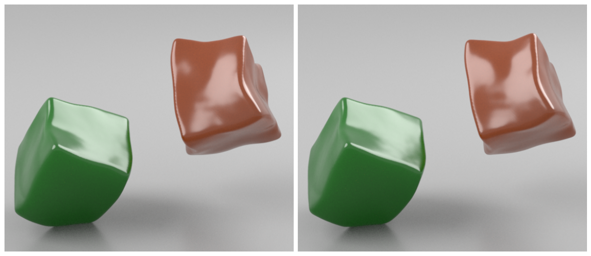

Fig. 14. **Comparison.** The result of our adaptive formulation (left) compared with regular MPM on a dense grid (right).

A. Bargteil, C. Wojtan, J. Hodgins, and G. Turk. 2007. A finite element method for animating large viscoplastic flow. *ACM Trans Graph* 26, 3 (2007).

C. Batty, S. Xenos, and B. Houston. 2010. Tetrahedral Embedded Boundary Methods for Accurate and Flexible Adaptive Fluids. In *Proc of Eurographics*.

J. Bonet and R. Wood. 2008. *Nonlinear continuum mechanics for finite element analysis*. Cambridge University Press.

J. Brackbill, D. Kothe, and H. Ruppel. 1988. FLIP: A low-dissipation, PIC method for fluid flow. *Comp Phys Comm* 48 (1988), 25–38.

R. Bridson. 2008. *Fluid simulation for Comp Graph*. Taylor & Francis.

S. Capell, S. Green, B. Curless, Tom D., and Z. Popović. 2002. A multiresolution framework for dynamic deformations. In *Proc of the 2002 ACM SIGGRAPH/Eurographics Symp on Comp Anim*. ACM, 41–47.

N. Chentanez and M. Müller. 2011. Real-time Eulerian water simulation using a restricted tall cell grid. *ACM Trans on Graph (TOG)* 30, 4 (2011), 82.

J. M. Cohen, S. Tariq, and S. Green. 2010. Interactive fluid-particle simulation using translating Eulerian grids. In *Proc of the 2010 ACM SIGGRAPH Symp on Interactive 3D Graph and Games*. ACM, 15–22.

N. Daphalapurkar, H. Lu, D. Coker, and R. Komanduri. 2007. Simulation of dynamic crack growth using the generalized interpolation material point (GIMP) method. *Int J Fract* 143, 1 (2007), 79–102.

G. Daviet and F. Bertails-Descoubes. 2016. A Semi-Implicit Material Point Method for the Continuum Simulation of Granular Materials. *ACM Trans Graph* 35, 4 (July 2016).

G. Debunne, M. Desbrun, A. Barr, and M-P. Cani. 1999. Interactive multiresolution Anim of deformable models. In *Comp Anim and SimulationâĂŽ99*. Springer, 133–144.

C. Dick, J. Georgii, and R. Westermann. 2011. A hexahedral multigrid approach for simulating cuts in deformable objects. *IEEE Trans on Vis and Comp Graph* 17, 11 (2011), 1663–1675.

R. E. English, L. Qiu, Y. Yu, and R. Fedkiw. 2013. Chimera grids for water simulation. In *Proc of the 12th ACM SIGGRAPH/Eurographics Symp on Comp Anim*. ACM, 85–94.

F. Ferstl, R. Ando, C. Wojtan, R. Westermann, and N. Thuerey. 2016. Narrow band FLIP for liquid simulations. In *Comp Graph Forum*, Vol. 35. Wiley Online Library, 225–232.

T. Fries, A. Byfut, A. Alizada, K. Cheng, and A. Schröder. 2011. Hanging nodes and XFEM. *Int J Numer Meth Eng* 86, 4-5 (2011), 404–430.

M. Gao, A. Pradhana, C. Jiang, and E. Sifakis. 2017. Supplemental Document: An Adaptive Generalized Interpolation Material Point Method for Simulating Elastoplastic Materials. (2017).

E. Grinspun, P. Krysl, and P. Schröder. 2002. CHARMS: a simple framework for adaptive simulation. *ACM Trans on Graph (TOG)* 21, 3 (2002), 281–290.

B. Houston, M. B. Nielsen, C. Batty, O. Nilsson, and K. Museth. 2006. Hierarchical RLE level set: A compact and versatile deformable surface representation. *ACM Trans on Graph (TOG)* 25, 1 (2006), 151–175.

T. Hughes. 2012. *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation.

G. Irving, E. Guendelman, F. Losasso, and R. Fedkiw. 2006. Efficient simulation of large bodies of water by coupling two and three dimensional techniques. In *ACM Trans on Graph (TOG)*, Vol. 25. ACM, 805–811.

C. Jiang, T. Gast, and J. Teran. 2017. Anisotropic elastoplasticity for cloth, knit and hair frictional contact. *ACM Trans Graph* 36, 4 (2017).

C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin. 2015. The Affine Particle-In-Cell Method. *ACM Trans Graph* 34, 4 (2015), 51:1–51:10.

C. Jiang, C. Schroeder, J. Teran, A. Stomakhin, and A. Selle. 2016. The Material Point Method for Simulating Continuum Materials. In *ACM SIGGRAPH 2016 Course*. 24:1–24:52.

P. Kaufmann, S. Martin, M. Botsch, and M. Gross. 2009. Flexible simulation of deformable models using discontinuous Galerkin FEM. *Graphical Models* 71, 4 (2009), 153–167.

G. Klár, T. Gast, A. Pradhana, C. Fu, C. Schroeder, C. Jiang, and J. Teran. 2016. Drucker-Prager Elastoplasticity for Sand Anim. *ACM Trans Graph* 35, 4 (July 2016).

F. Labelle and J. R. Shewchuk. 2007. Isosurface stuffing: fast tetrahedral meshes with good dihedral angles. In *ACM Trans on Graph (TOG)*, Vol. 26. ACM, 57.

G. Legrain, R. Allais, and P. Cartraud. 2011. On the use of the extended finite element method with quadtree/octree meshes. *Int J Numer Meth Eng* 86, 6 (2011), 717–743.

Y.P. Lian, P.F. Yang, X. Zhang, F. Zhang, Y. Liu, and P. Huang. 2015. A mesh-grading material point method and its parallelization for problems with localized extreme deformation. *Comp Meth App Mech Eng* 289 (2015), 291 – 315.

Y. Lian, X. Zhang, F. Zhang, and X. Cui. 2014. Tied interface grid material point method for problems with localized extreme deformation. *Int J Imp Eng* 70 (2014), 50–61.

H. Liu, N. Mitchell, M. Aanjaneya, and E. Sifakis. 2016. A scalable schur-complement fluids solver for heterogeneous compute platforms. *ACM Trans on Graph (TOG)* 35, 6 (2016), 201.

F. Losasso, F. Gibou, and R. Fedkiw. 2004. Simulating water and smoke with an octree data structure. In *ACM Trans Graph*, Vol. 23. ACM, 457–462.

J. Ma, H. Lu, and R. Komanduri. 2006. Structured mesh refinement in generalized interpolation material point (GIMP) method for simulation of dynamic problems. *Comp Model Eng & Sci* 12, 3 (2006), 213.

J. Ma, H. Lu, B. Wang, S. Roy, R. Hornung, A. Wissink, and R. Komanduri. 2005. Multiscale simulations using generalized interpolation material point (GIMP) method and SAMRAI parallel processing. *Comp Model Eng & Sci* 8, 2 (2005), 135–152.

P-L. Manteaux, C. Wojtan, R. Narain, S. Redon, F. Faure, and M-P. Cani. 2016. Adaptive physically based models in Comp Graph. In *Comp Graph Forum*. Wiley Online Library.

C. Mast. 2013. *Modeling landslide-induced flow interactions with structures using the Material Point Method*. Ph.D. Dissertation.

K. Museth. 2013. VDB: High-resolution sparse volumes with dynamic topology. *ACM Trans on Graph (TOG)* 32, 3 (2013), 27.

M. A. Otaduy, D. Germann, S. Redon, and M. Gross. 2007. Adaptive deformations with fast tight bounds. In *Proc of the 2007 ACM SIGGRAPH/Eurographics Symp on Comp Anim*. Eurographics Association, 181–190.

S. Patel, A. Chu, J. Cohen, and F. Pighin. 2005. Fluid simulation via disjoint translating grids. In *ACM SIGGRAPH 2005 Sketches*. ACM, 139.

D. Ram, T. Gast, C. Jiang, C. Schroeder, A. Stomakhin, J. Teran, and P. Kavehpour. 2015. A material point method for viscoelastic fluids, foams and sponges. In *Proc ACM SIGGRAPH/Eurographics Symp Comp Anim*. 157–163.

M. Seiler, D. Steinemann, J. Spillmann, and M. Harders. 2011. Robust interactive cutting based on an adaptive octree simulation mesh. *The Vis Comp* 27, 6-8 (2011), 519–529.

R. Setaluri, M. Aanjaneya, S. Bauer, and E. Sifakis. 2014. SPGrid: A Sparse Paged Grid Structure Applied to Adaptive Smoke Simulation. *ACM Trans Graph* 33, 6, Article 205 (Nov. 2014), 205:1–205:12 pages.

E. Sifakis and J. Barbic. 2012. FEM Simulation of 3D Deformable Solids: A Practitioner's Guide to Theory, Discretization and Model Reduction. In *ACM SIGGRAPH 2012 Courses (SIGGRAPH '12)*. Article 20, 50 pages.

E. Sifakis, T. Shinar, G. Irving, and R. Fedkiw. 2007. Hybrid simulation of deformable solids. In *Proc ACM SIGGRAPH/Eurographics Symp Comp Anim*. 81–90.

B. Solenthaler and M. Gross. 2011. Two-scale particle simulation. In *ACM Tran Graph*, Vol. 30. ACM, 81.

M. Steffen, R. M. Kirby, and M. Berzins. 2008. Analysis and reduction of quadrature errors in the material point method (MPM). *Int J Numer Meth Eng* 76, 6 (2008), 922–948.

D. Steinemann, M. A. Otaduy, and M. Gross. 2008. Fast adaptive shape matching deformations. In *Proc of the 2008 ACM SIGGRAPH/Eurographics Symp on Comp Anim*. Eurographics Association, 87–94.

A. Stomakhin, R. Howes, C. Schroeder, and J. Teran. 2012. Energetically consistent invertible elasticity. In *Proc Symp Comp Anim*. 25–32.

A. Stomakhin, C. Schroeder, L. Chai, J. Teran, and A. Selle. 2013. A Material Point Method for snow simulation. *ACM Trans Graph* 32, 4 (2013), 102:1–102:10.

A. Stomakhin, C. Schroeder, C. Jiang, L. Chai, J. Teran, and A. Selle. 2014. Augmented MPM for phase-change and varied materials. *ACM Trans Graph* 33, 4 (2014), 138:1–138:11.

D. Sulsky, S. Zhou, and H. Schreyer. 1995. Application of a particle-in-cell method to solid mechanics. *Comp Phys Comm* 87, 1 (1995), 236–252.

A. P. Tampubolon, T. Gast, G. Klár, C. Fu, J. Teran, C. Jiang, and K. Museth. 2017. Multi-species simulation of porous sand and water mixtures. *ACM Trans Graph* 36, 4 (2017).

H. Tan and J. A. Nairn. 2002. Hierarchical, adaptive, material point method for dynamic energy release rate calculations. *Comp Meth App Mech Eng* 191, 19âĂŞ20 (2002), 2123 – 2137.

M. Wicke, D. Ritchie, B. Klingner, S. Burke, J. Shewchuk, and J. O'Brien. 2010. Dynamic local remeshing for elastoplastic simulation. *ACM Trans Graph* 29, 4 (2010), 49:1–11.

C. Wojtan and G. Turk. 2008. Fast viscoelastic behavior with thin features. *ACM Trans Graph* 27, 3 (2008), 1–8.

Y. Yue, B. Smith, C. Batty, C. Zheng, and E. Grinspun. 2015. Continuum foam: a material point method for shear-dependent flows. *ACM Trans Graph* 34, 5 (2015), 160:1–160:20.

Y. Zhu and R. Bridson. 2005. Animating sand as a fluid. *ACM Trans Graph* 24, 3 (2005), 965–972.