UNIVERSITY OF CALIFORNIA

Los Angeles

A Material Point Method for Elastoplasticity with Ductile Fracture and Frictional Contact

A dissertation submitted in partial satisfaction of the requirements for the degree Doctor of Philosophy in Mathematics

by

Stephanie Wang

2020

ABSTRACT OF THE DISSERTATION

A Material Point Method for Elastoplasticity
with Ductile Fracture and Frictional Contact

by

Stephanie Wang
Doctor of Philosophy in Mathematics
University of California, Los Angeles, 2020
Professor Joseph M. Teran, Chair

Simulating physical materials with dynamic movements to photo-realistic resolution has always been one of the most crucial and challenging topics in Computer Graphics. This dissertation considers large-strain elastoplasticity theory applied to the low-to-medium stiffness regime, with topological changes and codimensional objects incorporated. We introduce improvements to the Material Point Method (MPM) for two particular objectives, simulating fracturing ductile materials and incorporation of MPM and Lagrangian Finite Element Method (FEM).

Our first contribution, simulating ductile fracture, utilizes traditional particle-based MPM [SSC13, SCS94] as well as the Lagrangian energy formulation of [JSS15] which uses a tetrahedron mesh, rather than particle-based estimation of the deformation gradient and potential energy. We model failure and fracture via elastoplasticity with damage. The material is elastic until its deformation exceeds a Rankine or von Mises yield condition. At that point, we use a softening model that shrinks the yield surface until it reaches the damage threshold. Once damaged, the material Lamé coefficients are modified to represent failed material.

This approach to simulating ductile fracture with MPM is successful, as MPM naturally captures the topological changes coming from the fracture. However, rendering the crack surfaces can be challenging. We design a novel visualization technique dedicated to rendering the material's boundary and its intersection with the evolving crack surfaces. Our approach uses a simple and efficient element splitting strategy for tetrahedron meshes to create crack surfaces. It employs an extrapolation technique based on the MPM simulation. For traditional particle-based MPM, we use an initial Delaunay tetrahedralization to connect randomly sampled MPM particles. Our visualization technique is a post-process and can run after the MPM simulation for efficiency. We demonstrate our method with several challenging simulations of ductile failure with considerable and persistent self-contact and applications with thermomechanical models for baking and cooking.

Our second contribution, hybrid MPM–Lagrangian-FEM, aims to simulate elastic objects like hair, rubber, and soft tissues. It utilizes a Lagrangian mesh for internal force computation and a Eulerian grid for self-collision, as well as coupling with external materials. While recent MPM techniques allow for natural simulation of hyperelastic materials represented with Lagrangian meshes, they utilize an updated Lagrangian discretization and use the Eulerian grid degrees of freedom to take variations of the potential energy. It often coarsens the degrees of freedom of the Lagrangian mesh and can lead to artifacts. We develop a hybrid approach that retains Lagrangian degrees of freedom while still allowing for natural coupling with other materials simulated with traditional MPM, e.g., sand, snow, etc. Furthermore, while recent MPM advances allow for resolution of frictional contact with codimensional simulation of hyperelasticity, they do not generalize to the case of volumetric materials. We show that our hybrid approach resolves these issues. We demonstrate the efficacy of our technique with examples that involve elastic soft tissues coupled with kinematic skeletons, extreme deformation, and coupling with various elastoplastic materials. Our approach also naturally allows for two-way rigid body coupling.

The dissertation of Stephanie Wang is approved.

Jeffrey D. Eldredge

Wotao Yin

Luminita Aura Vese

Joseph M. Teran, Committee Chair

University of California, Los Angeles

2020

To Năi nai.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

My highest appreciation goes to my advisor, Professor Joseph M. Teran. He participated in every crunch, guiding me to embrace the wildest vision for research while breaking down problems to the finest details. His kindness and guidance led me through the nebulous path to becoming a researcher.

I thank Professor Jeff D. Eldredge, Professor Luminita Aura Vese, and Professor Wotao Yin for serving on my dissertation committee and providing valuable advice for improving and extending my work.

I owe Professor Wilfrid Gangbo special thanks for his support and advice during my last year. I cherished our illuminating math discussions and the chance to learn the theories that I always craved to understand.

I thank Professor Inwon C. Kim, Professor Mario Bonk, Professor Lieven Vandenberghe, Professor Ko Honda, and all of my teachers at UCLA for their inspiring teaching. Their lessons were precious to me and have become the building blocks of my mathematical worldview. I also thank Professor Mason A. Porter for his mentoring and advice on our joint work.

I am grateful for my time spent working at Walt Disney Animation Studios over the summer of 2018. I thank my mentors, Mark McLaughlin and Alex McAdams, my referrer, Alexey Stomakhin, my fun and incredibly acknowledgeable colleagues, Noel Villegas, Jessie Erickson, and John Kosnik, and my extremely proficient collaborator, Rasmus Thamdorf. Their passion and expertise in computer animation are awe-inspiring.

I thank Professor Johan Gaume and all the members of SLAB for the great summer I spent in Switzerland in 2019. I appreciate our friendship and the productive workspace we had. I will always remember my time there fondly.

I can't express enough gratitude for my collaborators and mentors, Theodore F. Gast, Jiang Chenfanfu, and Matt Jacobs, for their tutelage and teaching. They taught me the crucial skills of thinking like a researcher and attacking problems with diligence.

I want to thank my chosen family and friends, who cared for and supported me when my family is out of reach. I couldn't have made it without the support of Chayan Chang, Serj Kazar, and many more. May our friendships last forever.

I have to thank Patrick Yi-Ping Lin, who purely unintentionally inspired me to pursue mathematics at the very beginning.

I thank my parents for everything they did for me, starting before day zero. I thank my dad for supporting me and caring for me all these years. Finally, I thank my mother, who deserves all the credit for inspiring me to be more than what I can be.

VITA

2009–2013	B.S. (Mathematics), National Taiwan University.
2015–2020	Teaching Assistant, Math Department, UCLA.
2016–2020	Research Assistant, Math Department, UCLA.
2016	M.S. (Mathematics), UCLA.
2018	Summer Technology Intern, Walt Disney Animation Studios.
2019	Principal Instructor, Math Department, UCLA.
2019	Summer Visiting Scholar, École Polytechnique Fédérale de Lausanne, Switzerland

PUBLICATIONS

S. Wang, M. Ding, T. Gast, L. Zhu, S. Gagniere, C. Jiang, J. Teran, Simulation and Visualization of Ductile Fracture with the Material Point Method, *Proc ACM on Computer Graphics and Interactive Techniques, 2(2)*, pp. 18, 2019 (SCA2019 Best Paper Award)

X. Han, T. Gast, Q. Guo, S. Wang, C. Jiang, J. Teran, A Hybrid Material Point Method for Frictional Contact with Diverse Materials, *Proc ACM on Computer Graphics and Interactive Techniques, 2(2)*, pp. 17, 2019

J. Carlen, J. Pont, C. Mentus, S. Chang, S. Wang, M. Porter, Role Detection in Bicycle-Sharing Networks Using Multilayer Stochastic Block Models, arXiv:1908.09440

M. Ding, X. Han, S. Wang, T. Gast, J. Teran, A thermomechanical material point method for baking and cooking, ACM Transactions on Graphics (SIGGRAPH Asia 2019, 38(6), 192, 2019

CHAPTER 1

Introduction

1.1 Contributions

This dissertation consists of two major components: fracture and frictional contact, originally developed in [WDG19b] and [HGG19], respectively. The key contributions are listed below:

1. Fracture

    - To our knowledge, we are the first to create an MPM-compliant meshing technique that admits texturing of the surface and provides a sharp finish for fracturing objects. It provides comparable visual details for simulations with one-tenth of the particle count.
    - Our meshing technique works with any MPM simulations (for example, thermo-mechanical models for baking).
    - By design, a post-process meshing method saves computational resources as users need not call the post-process unless the simulated results pass preliminary quality tests.
    - A particle-based MPM framework for simulating material fracture without requiring quality mesh, which is often the bottleneck for mesh-based methods.
    - A Lagrangian MPM framework for simulating material fracture, which prevents numerical fracture.

2. Frictional Contact

- A novel method for two-way-coupling of MPM and Lagrangian FEM.
- Our approach also provides a new method for handling self-collision for Lagrangian FEM using an Eulerian grid.
- A plastic model for modeling the frictional contact between MPM particles and Lagrangian meshes.
- A new approach for coupling rigid body objects with MPM materials using Lagrangian FEM

1.2 Dissertation overview

Chapter 2

We select and review parts of continuum mechanics that lead to the governing equations of elastoplasticity. We cover Eulerian and Lagrangian dynamics, hyperelasticity, multiplicative decomposition wand large-strain models, yield surfaces, and associative plastic projection. We expect readers to have basic knowledge in mechanics, differential equations, and preferably some differential geometry and convex analysis to follow the derivation.

Chapter 3

We present our work in the simulation and visualization of ductile fracture from [WDG19b]. We begin by reviewing previous work that aims to solve the fracture problem with various approaches. We discuss the particular elastoplastic model used to simulate fracturing material and the numerical discretization used for computation. We then demonstrate the visualization method, consisting of three processes—processing the topology, smoothing the crack surface, and extrapolating positions for additional vertices. We provide extensive examples (e.g. Figure 3.1, 3.3, etc) for our method.

We also supply several results from [DHW19] to show the versatility of our mesh-processing technique developed for fracture (e.g. Figure 3.19, etc).

Chapter 4

We present our work with the hybrid MPM for frictional contact with diverse materials from [HGG19]. We provide a detailed explanation of our integrated simulation engine that combines MPM and Lagrangian FEM. We use quadrature particles sampled on the boundary of the meshed objects to achieve two-way-coupling. Detailed explanation of the modeling of frictional contact and impulse exchange are provided in Section 4.4.4 and Section 4.5.3. We then discuss the two-way-coupling with rigid body and traditional MPM. We provide extensive examples (e.g. Figure 4.1, 4.2, etc) for our method.

CHAPTER 2

Continuum Mechanics

Continuum mechanics is the foundation of all governing equations this dissertation concerns. The Material Point Method itself is a hybrid Eulerian and Lagrangian numerical solver, and the hybrid MPM–Lagrangian-FEM requires Lagrangian dynamics as well. We go over the derivation of hyperelasticity in both Eulerian and Lagrangian view in Section 2.1 and 2.2. Some of the material in this chapter has been published in [WDG19a].

2.1 Eulerian dynamics

We follow mostly [Tao18] to derive the continuity equation and Cauchy's momentum equation while carefully identifying the physical laws and mathematical assumptions behind them. Although these assumptions apply to most cases studied in this dissertation, it is essential to note that other different types of material often agree with most of these assumptions except a few.

In this section, we work with the Eulerian view—the mathematical functions used to approximate physical quantities are defined on *the set of deformed space.* We will further clarify the distinction between Eulerian and Lagrangian views in section 2.2.

2.1.1 Newton's laws

Consider a set of N particles $\{P^{(a)}\}_{a=1}^N$ moving in space-time $\mathbb{R} \times \mathbb{R}^3$. Each of them has mass $m^{(a)} > 0$ and a trajectory $\mathbf{x}^{(a)} : \mathbb{R} \to \mathbb{R}^3$. (We do not consider the scenario where the mass of particles changes with time.) The force each particle experiences at time t is denoted as $\mathbf{F}^{(a)}(t) \in \mathbb{R}^3$. We invoke the first physical law,

Newton's Second Law

$$\mathbf{F} = m\mathbf{a}. \tag{2.1}$$

This gives the equation of motion,

$$m^{(a)}\ddot{\mathbf{x}}^{(a)}(t) = \mathbf{F}^{(a)}(t). \tag{2.2}$$

Here the double dot denotes the second time-derivative.

Assuming **forces are additive**, we separately consider the external force (e.g. gravity) and internal force (force induced by interaction amongst particles $\{P^{(a)}\}_{a=1}^N$),

$$\mathbf{F}^{(a)} = \mathbf{F}^{(a)}_{ext} + \mathbf{F}^{(a)}_{int}. \tag{2.3}$$

Assuming **only pair-wise interactions are significant**, that is, neglecting the interaction between, say, triplets $(P^{(a)}, P^{(b)}, P^{(c)})$ or more, we can write down the force decomposition,

$$\mathbf{F}^{(a)} = \mathbf{F}^{(a)}_{ext} + \sum_{b=1}^{N} \mathbf{F}^{(ab)}, \tag{2.4}$$

where $\mathbf{F}^{(ab)}$ denotes the interaction force exerted on $P^{(a)}$ by $P^{(b)}$. The following two laws can come in handy.

Newton's First Law

$$\mathbf{F}^{(aa)} = 0. \tag{2.5}$$

Newton's Third Law

$$\mathbf{F}^{(ab)} = -\mathbf{F}^{(ba)}. \tag{2.6}$$

The equation of motion is now

$$m^{(a)}\ddot{\mathbf{x}}^{(a)}(t) = \mathbf{F}^{(a)}_{ext} + \sum_{b:b\neq a} \mathbf{F}^{(ab)}. \tag{2.7}$$

From Newton's third law (2.6), we can deduce that the total (linear) momentum of the system $\mathbf{L}(t) = \sum_{a=1}^{N} m^{(a)}\dot{\mathbf{x}}^{(a)}(t)$ remains constant if there's no external force.

$$\begin{aligned}
\frac{d}{dt}\mathbf{L}(t) &= \frac{d}{dt}\left(\sum_{a=1}^{N} m^{(a)}\dot{\mathbf{x}}^{(a)}(t)\right) \\
&= \sum_{a=1}^{N} m^{(a)}\ddot{\mathbf{x}}^{(a)}(t) \\
&= \sum_{a=1}^{N}\sum_{b:b\neq a} \mathbf{F}^{(ab)} \\
&= \sum_{a,b:a<b} \mathbf{F}^{(ab)} + \sum_{a,b:b<a} \mathbf{F}^{(ab)} \\
&= \sum_{a,b:a<b} (\mathbf{F}^{(ab)} - \mathbf{F}^{(ab)}) = 0.
\end{aligned}$$

Another assumption we make is that the interaction force $\mathbf{F}^{(ab)}$ must be parallel to the displacement $\mathbf{x}^{(a)} - \mathbf{x}^{(b)}$, or,

Vanishing torque

$$(\mathbf{x}^{(a)} - \mathbf{x}^{(b)}) \wedge \mathbf{F}^{(ab)} = 0. \tag{2.8}$$

We can show that the total angular momentum of the system $\mathbf{J}(t) = \sum_{a=1}^{N} m^{(a)}\mathbf{x}^{(a)} \wedge \dot{\mathbf{x}}^{(a)}$ remains constant if there's no external force.

$$\begin{aligned}
\frac{d}{dt}\mathbf{J}(t) &= \frac{d}{dt}\left(\sum_{a=1}^{N} m^{(a)}\mathbf{x}^{(a)} \wedge \dot{\mathbf{x}}^{(a)}\right) \\
&= \sum_{a=1}^{N} m^{(a)}\left(\dot{\mathbf{x}}^{(a)} \wedge \dot{\mathbf{x}}^{(a)} + \mathbf{x}^{(a)} \wedge \ddot{\mathbf{x}}^{(a)}\right) \\
&= \sum_{a=1}^{N} \mathbf{x}^{(a)} \wedge (m^{(a)}\ddot{\mathbf{x}}^{(a)}) \\
&= \sum_{a=1}^{N} \mathbf{x}^{(a)} \wedge \left(\sum_{b:b\neq a} \mathbf{F}^{(ab)}\right) \\
&= \sum_{a,b:a<b} \mathbf{x}^{(a)} \wedge \mathbf{F}^{(ab)} + \sum_{a,b:b<a} \mathbf{x}^{(b)} \wedge \mathbf{F}^{(ab)} \\
&= \sum_{a,b:a<b} (\mathbf{x}^{(a)} - \mathbf{x}^{(b)}) \wedge \mathbf{F}^{(ab)} = 0.
\end{aligned}$$

Note the above definition is the angular momentum with regard to the origin $\mathbf{x} = 0$. By the following formula we can deduce that the angular momentum with regard to any other anchor point $\mathbf{x}_{anchor}$ also remains constant if there's no external force.

$$\begin{aligned}
\mathbf{J}(t; \mathbf{x}_{anchor}) &= \sum_{a=1}^{N} m^{(a)}(\mathbf{x}^{(a)} - \mathbf{x}_{anchor}) \wedge \dot{\mathbf{x}}^{(a)} \\
&= \sum_{a=1}^{N} m^{(a)}\mathbf{x}^{(a)} \wedge \dot{\mathbf{x}}^{(a)} - \mathbf{x}_{anchor} \wedge \left(\sum_{a=1}^{N} m^{(a)}\dot{\mathbf{x}}^{(a)}\right) \\
&= \mathbf{J}(t) - \mathbf{x}_{anchor} \wedge \mathbf{L}(t).
\end{aligned}$$

2.1.2 Continuity equation

As the number of particles N approaches infinity (or more precisely, the number of particles per unit volume is comparable to Avagadro's constant), it is worth considering mathematical functions defined on the continuum that approximate the physical quantities of interests. The two most important physical quantities of a moving body are mass and velocity (momentum).

The discrete measure of the mass distribution is given by

$$\mu_{mass}(t) = \sum_{a=1}^{N} m^{(a)} \delta_{\mathbf{x}^{(a)}(t)}. \tag{2.9}$$

The discrete measure of the momentum distribution is given by

$$\mu_{momentum}(t) = \sum_{a=1}^{N} m^{(a)} \dot{\mathbf{x}}^{(a)}(t) \delta_{\mathbf{x}^{(a)}(t)}. \tag{2.10}$$

We assume **there exists a function $\rho : \mathbb{R} \times \mathbb{R}^3 \to \mathbb{R}^+$ that approximates μ_{mass},** that is,

$$\mu_{mass}(t) \approx \rho(t, \mathbf{x}) d\mathcal{L}(\mathbf{x}), \tag{2.11}$$

where $d\mathcal{L}$ denotes the Lebesgue measure of $\mathbb{R}^3$. This approximation is articulated by testing with smooth, compactly-supported functions,

$$\forall \psi \in C_c^\infty(\mathbb{R} \times \mathbb{R}^3), \int_{\mathbb{R}} \sum_{a=1}^{N} m^{(a)} \psi(t, \mathbf{x}^{(a)}(t)) dt \approx \int_{\mathbb{R}} \int_{\mathbb{R}^3} \rho(t, \mathbf{x}) \psi(t, \mathbf{x}) d\mathbf{x} dt. \tag{2.12}$$

An immediate result is that the total mass of the system remains constant over time,

$$\frac{d}{dt} \int_{\mathbb{R}} \rho(t, \mathbf{x}) d\mathbf{x} = \frac{d}{dt} \left(\sum_{a=1}^{N} m^{(a)} \right) = 0. \tag{2.13}$$

We also assume **there exists a function $\mathbf{u} : \mathbb{R} \times \mathbb{R}^3 \to \mathbb{R}^3$ that approximates velocity**, that is,

$$\mu_{momentum}(t) = \sum_{a=1}^{N} m^{(a)} \dot{\mathbf{x}}^{(a)}(t) \delta_{\mathbf{x}^{(a)}(t)} \approx \rho(t, \mathbf{x}) \mathbf{u}(t, \mathbf{x}) d\mathcal{L}(\mathbf{x}). \tag{2.14}$$

This is similarly articulated by

$$\forall \psi \in C_c^\infty(\mathbb{R} \times \mathbb{R}^3), \int_{\mathbb{R}} \sum_{a=1}^{N} m^{(a)} \dot{\mathbf{x}}^{(a)}(t) \psi(t, \mathbf{x}^{(a)}(t)) dt \approx \int_{\mathbb{R}} \int_{\mathbb{R}^3} \rho(t, \mathbf{x}) \mathbf{u}(t, \mathbf{x}) \psi(t, \mathbf{x}) d\mathbf{x} dt. \tag{2.15}$$

Note that any smoothness of $\mathbf{u} : \mathbb{R} \times \mathbb{R}^3 \to \mathbb{R}^3$ would assume that particles in proximity must have similar velocity. Boltzmann equation also aims to describe moving particles but does not build on this assumption. It focuses on stochastic behavior, which is not concerned in this dissertation.

For $\psi \in C_c^\infty(\mathbb{R} \times \mathbb{R}^3)$, due to its compact support (in time), we always have

$$\int_{\mathbb{R}} \frac{d}{dt} \int_{\mathbb{R}^3} \psi(t, \mathbf{x}^{(a)}(t)) d\mu_{mass}(t)(\mathbf{x}) dt = 0 = \int_{\mathbb{R}} \frac{d}{dt} \sum_{a=1}^{N} m^{(a)} \psi(t, \mathbf{x}^{(a)}(t)) dt. \tag{2.16}$$

If we focus on a single particle $a \in \{1, \cdots, N\}$, viewing $\psi(t, \mathbf{x}^{(a)}(t))$ as a function in time,

$$\begin{aligned} \frac{d}{dt}\psi(t, \mathbf{x}^{(a)}(t)) &= \frac{\partial \psi}{\partial t}(t, \mathbf{x}^{(a)}(t)) + \frac{\partial \psi}{\partial \mathbf{x}}(t, \mathbf{x}^{(a)}(t)) \cdot \frac{d}{dt}\mathbf{x}^{(a)}(t) \\ &= (\partial_t + \mathbf{u} \cdot \partial_{\mathbf{x}})\psi(t, \mathbf{x}^{(a)}(t)). \end{aligned} \tag{2.17}$$

The combination $D_t := \partial_t + \mathbf{u} \cdot \partial_{\mathbf{x}}$ is called the *material derivative.* The fact that $\mu_{mass}(t)$ are made out of delta measures concentrating on moving positions $\{\mathbf{x}^{(a)}(t)\}_{a=1}^N$ rewrites the seemingly harmless equation (2.16),

$$\begin{aligned} 0 &= \int_{\mathbb{R}} \frac{d}{dt} \sum_{a=1}^{N} m^{(a)} \psi(t, \mathbf{x}^{(a)}(t)) dt \\ &= \int_{\mathbb{R}} \sum_{a=1}^{N} m^{(a)} \frac{d}{dt}\psi(t, \mathbf{x}^{(a)}(t)) dt \\ &= \int_{\mathbb{R}} \sum_{a=1}^{N} m^{(a)} (\partial_t + \mathbf{u} \cdot \partial_{\mathbf{x}})\psi(t, \mathbf{x}^{(a)}(t)) dt && \text{(by (2.17))} \\ &= \int_{\mathbb{R}} \int_{\mathbb{R}^3} \rho(t, \mathbf{x})(\partial_t + \mathbf{u} \cdot \partial_{\mathbf{x}})\psi(t, \mathbf{x}) d\mathbf{x} dt && \text{(mass approximation (2.11))} \\ &= \int_{\mathbb{R}} \int_{\mathbb{R}^3} \psi(t, \mathbf{x})(-\partial_t \rho - \nabla_{\mathbf{x}} \cdot (\rho \mathbf{u}))(t, \mathbf{x}) d\mathbf{x} dt. \end{aligned}$$

Since the test function ψ is arbitrary, we conclude that

$$\partial_t \rho + \nabla_{\mathbf{x}} \cdot (\rho \mathbf{u}) = 0. \tag{2.18}$$

This is called the *continuity equation.* As we can see, this is merely a result from assuming that ρ and $\mathbf{u}$ approximate mass and velocity, respectively.

2.1.3 Cauchy's momentum equation and Cauchy's stress tensor $\boldsymbol{\sigma}$

Using the same thought as in (2.16) with $\mu_{momentum}$, we get

$$\int_{\mathbb{R}} \frac{d}{dt} \int_{\mathbb{R}^3} \psi(t, \mathbf{x}^{(a)}(t)) d\mu_{momentum}(t)(\mathbf{x}) dt = 0 = \int_{\mathbb{R}} \frac{d}{dt} \sum_{a=1}^{N} m^{(a)} \dot{\mathbf{x}}^{(a)}(t) \psi(t, \mathbf{x}^{(a)}(t)) dt. \quad (2.19)$$

Expand the right-hand-side and invoke the equation of motion (2.2),

$$\begin{aligned} \frac{d}{dt} \left(\dot{\mathbf{x}}^{(a)}(t) \psi(t, \mathbf{x}^{(a)}(t)) \right) &= m^{(a)} \ddot{\mathbf{x}}^{(a)}(t) \psi(t, \mathbf{x}^{(a)}(t)) + m^{(a)} \dot{\mathbf{x}}^{(a)}(t) D_t \psi(t, \mathbf{x}^{(a)}(t)) \\ &= \mathbf{F}^{(a)}(t) \psi(t, \mathbf{x}^{(a)}(t)) + m^{(a)} (\mathbf{u} D_t \psi)(t, \mathbf{x}^{(a)}(t)). \end{aligned}$$

In particular, using the force decomposition in (2.7),

$$\begin{aligned} \sum_{a=1}^{N} \mathbf{F}^{(a)}(t) \psi(t, \mathbf{x}^{(a)}(t)) &= \sum_{a=1}^{N} \mathbf{F}^{(a)}_{ext}(t) \psi(t, \mathbf{x}^{(a)}(t)) + \sum_{a,b: a \neq b} \mathbf{F}^{(ab)}(t) \psi(t, \mathbf{x}^{(a)}(t)) \\ \sum_{a,b: a \neq b} \mathbf{F}^{(ab)}(t) \psi(t, \mathbf{x}^{(a)}(t)) &= \frac{1}{2} \sum_{a,b: a \neq b} \mathbf{F}^{(ab)}(t) \left(\psi(t, \mathbf{x}^{(a)}(t)) - \psi(t, \mathbf{x}^{(b)}(t)) \right) \\ &= \frac{1}{2} \sum_{a,b: a \neq b} \mathbf{F}^{(ab)}(t) \left((\mathbf{x}^{(a)}(t) - \mathbf{x}^{(b)}(t)) \cdot \nabla_{\mathbf{x}} \psi(t, \mathbf{x}^{(a)}(t)) + \mathcal{O}(|\mathbf{x}^{(a)}(t) - \mathbf{x}^{(b)}(t)|^2) \right) \\ &= \frac{1}{2} \sum_{a,b: a \neq b} \mathbf{F}^{(ab)}(t) (\mathbf{x}^{(a)}(t) - \mathbf{x}^{(b)}(t)) \cdot \nabla_{\mathbf{x}} \psi(t, \mathbf{x}^{(a)}(t)) \\ &\quad + \frac{1}{2} \sum_{a,b: a \neq b} \mathbf{F}^{(ab)}(t) \mathcal{O}(|\mathbf{x}^{(a)}(t) - \mathbf{x}^{(b)}(t)|^2). \end{aligned}$$

We assume **short-range interactions**, that is, we assume force $\mathbf{F}^{(ab)}(t)$ is only significant for $\mathbf{x}^{(a)}(t)$ very close to $\mathbf{x}^{(b)}(t)$, thus we can discard the second term in the above equation.

We rewrite (2.19) with this approximation

$$\begin{aligned}
0 &= \int_{\mathbb{R}} \frac{d}{dt} \sum_{a=1}^{N} m^{(a)} \dot{\mathbf{x}}^{(a)}(t) \psi(t, \mathbf{x}^{(a)}(t)) dt \\
&= \int_{\mathbb{R}} \sum_{a=1}^{N} \mathbf{F}^{(a)}(t) \psi(t, \mathbf{x}^{(a)}(t)) dt + \int_{\mathbb{R}} \sum_{a=1}^{N} m^{(a)} (\mathbf{u} D_t \psi)(t, \mathbf{x}^{(a)}(t)) dt \\
&= \int_{\mathbb{R}} \sum_{a=1}^{N} \mathbf{F}^{(a)}(t) \psi(t, \mathbf{x}^{(a)}(t)) dt + \int_{\mathbb{R}} \int_{\mathbb{R}^3} \rho(t, \mathbf{x}) (\mathbf{u} D_t \psi)(t, \mathbf{x}) d\mathbf{x} dt \\
&= \int_{\mathbb{R}} \sum_{a=1}^{N} \mathbf{F}_{ext}^{(a)}(t) \psi(t, \mathbf{x}^{(a)}(t)) dt + \int_{\mathbb{R}} \frac{1}{2} \sum_{a,b:a \neq b} \mathbf{F}^{(ab)}(t) (\mathbf{x}^{(a)}(t) - \mathbf{x}^{(b)}(t)) \cdot \nabla_{\mathbf{x}} \psi(t, \mathbf{x}^{(a)}(t)) dt \\
&\qquad + \int_{\mathbb{R}} \int_{\mathbb{R}^3} \rho(t, \mathbf{x}) (\mathbf{u} D_t \psi)(t, \mathbf{x}) d\mathbf{x} dt.
\end{aligned}$$

Define the stress tensor

$$\boldsymbol{\Sigma}^{(a)}(t) := -\frac{1}{2} \sum_{b:b \neq a} \mathbf{F}^{(ab)}(t) (\mathbf{x}^{(a)}(t) - \mathbf{x}^{(b)}(t))^T. \tag{2.20}$$

Note that this is a *second order tensor* (a matrix), it takes vectors as input (displacement $\mathbf{x}^{(a)} - \mathbf{x}^{(b)}$) and outputs vectors (force $\mathbf{F}^{(ab)}$). We will discuss more on tensors in section 2.2.2. We rewrite the equation with $\boldsymbol{\Sigma}^{(a)}$,

$$\begin{aligned}
0 = \int_{\mathbb{R}} \sum_{a=1}^{N} \mathbf{F}_{ext}^{(a)}(t) \psi(t, \mathbf{x}^{(a)}(t)) dt - \int_{\mathbb{R}} \sum_{a=1}^{N} \boldsymbol{\Sigma}^{(a)}(t) \nabla_{\mathbf{x}} \psi(t, \mathbf{x}^{(a)}(t)) dt \\
+ \int_{\mathbb{R}} \int_{\mathbb{R}^3} \rho(t, \mathbf{x}) (\mathbf{u} D_t \psi)(t, \mathbf{x}) d\mathbf{x} dt.
\end{aligned}$$

Much like we assumed the existence of approximation ρ and $\mathbf{u}$ for discrete quantities like mass and velocity, we assume **there exists a matrix-valued function $\boldsymbol{\sigma} : \mathbb{R} \times \mathbb{R}^3 \to M(3, \mathbb{R})$ that approximates the stress tensor**. We articulate this approximation by testing it with compactly supported functions,

$$\forall \psi \in C_c^{\infty}(\mathbb{R} \times \mathbb{R}^3), \int_{\mathbb{R}} \sum_{a=1}^{N} \boldsymbol{\Sigma}^{(a)}(t) \psi(t, \mathbf{x}^{(a)}(t)) dt \approx \int_{\mathbb{R}} \int_{\mathbb{R}^3} \boldsymbol{\sigma}(t, \mathbf{x}) \psi(t, \mathbf{x}) d\mathbf{x} dt. \tag{2.21}$$

We also assume **there exists a function $\mathbf{b} : \mathbb{R} \times \mathbb{R}^3 \to \mathbb{R}^3$ that approximates the external forces (body force) on the particles**, that is,

$$\forall \psi \in C_c^\infty(\mathbb{R} \times \mathbb{R}^3), \int_{\mathbb{R}} \sum_{a=1}^{N} \mathbf{F}_{ext}^{(a)}(t)\psi(t, \mathbf{x}^{(a)}(t))dt \approx \int_{\mathbb{R}} \int_{\mathbb{R}^3} \mathbf{b}(t, \mathbf{x})\psi(t, \mathbf{x})d\mathbf{x}dt. \tag{2.22}$$

The most common type of body force is gravity, which is a constant function in time and space $\mathbf{b}(t, \mathbf{x}) \equiv \mathbf{g}$.

Finally, we summarize the above derivation

$$\begin{aligned}
0 &= \int_{\mathbb{R}} \sum_{a=1}^{N} \mathbf{F}_{ext}^{(a)}(t)\psi(t, \mathbf{x}^{(a)}(t))dt - \int_{\mathbb{R}} \sum_{a=1}^{N} \mathbf{\Sigma}^{(a)}(t)\nabla_{\mathbf{x}}\psi(t, \mathbf{x}^{(a)}(t))dt + \int_{\mathbb{R}} \int_{\mathbb{R}^3} \rho(t, \mathbf{x})(\mathbf{u}D_t\psi)(t, \mathbf{x})d\mathbf{x}dt \\
&= \int_{\mathbb{R}} \int_{\mathbb{R}^3} \mathbf{b}(t, \mathbf{x})\psi(t, \mathbf{x})d\mathbf{x}dt - \int_{\mathbb{R}} \int_{\mathbb{R}^3} \boldsymbol{\sigma}(t, \mathbf{x})\nabla_{\mathbf{x}}\psi(t, \mathbf{x})d\mathbf{x}dt + \int_{\mathbb{R}} \int_{\mathbb{R}^3} \rho(t, \mathbf{x})(\mathbf{u}D_t\psi)(t, \mathbf{x})d\mathbf{x}dt \\
&= \int_{\mathbb{R}} \int_{\mathbb{R}^3} \mathbf{b}(t, \mathbf{x})\psi(t, \mathbf{x})d\mathbf{x}dt + \int_{\mathbb{R}} \int_{\mathbb{R}^3} \nabla_{\mathbf{x}} \cdot \boldsymbol{\sigma}(t, \mathbf{x})\psi(t, \mathbf{x})d\mathbf{x}dt + \int_{\mathbb{R}} \int_{\mathbb{R}^3} D_t^*(\rho\mathbf{u})(t, \mathbf{x})\psi(t, \mathbf{x})d\mathbf{x}dt.
\end{aligned}$$

Here D_t^* denotes the adjoint of the material derivative D_t, that is

$$\int_{\mathbb{R}} \int_{\mathbb{R}^3} fD_t\psi d\mathbf{x}dt = \int_{\mathbb{R}} \int_{\mathbb{R}^3} (D_t^* f)\psi d\mathbf{x}dt \tag{2.23}$$

Using the Leibniz rule of D_t, we have that

$$\begin{aligned}
\int_{\mathbb{R}} \int_{\mathbb{R}^3} D_t^*(fg)\psi &= \int_{\mathbb{R}} \int_{\mathbb{R}^3} fgD_t\psi \\
&= \int_{\mathbb{R}} \int_{\mathbb{R}^3} f(D_t(g\psi) - (D_tg)\psi) \\
&= \int_{\mathbb{R}} \int_{\mathbb{R}^3} (D_t^* f)g\psi - f(D_tg)\psi
\end{aligned}$$

and $D_t^*(\rho\mathbf{u}) = (D_t^*\rho)\mathbf{u} - \rho D_t\mathbf{u}$. Here

$$D_t^*\rho = -\partial_t\rho + \mathbf{u} \cdot \nabla_{\mathbf{x}}\rho = 0,$$

since $D_t^*\rho$ is exactly as in the continuity equation. Since the test function ψ is arbitrary, we are now left with

$$\rho D_t\mathbf{u} = \nabla_{\mathbf{x}}\boldsymbol{\sigma} + \mathbf{b}. \tag{2.24}$$

This is called the *Cauchy's momentum equation.* We will reinterpret this equation with the Lagrangian view in section 2.2.

Remarks

Note that the introduction of a matrix-valued variable $\boldsymbol{\sigma}$ introduces $3 \times 3 = 9$ more unknowns. (The external force $\mathbf{b}$ is often coming from gravity.) The continuity equation and Cauchy's momentum equation alone make up $1 + 3$ equations. We will need more physical insights to derive equations for $\boldsymbol{\sigma}$. These equations are usually called the *constitutive equation.* Nonetheless, we can use the vanishing torque assumption (2.8) (force is parallel to displacement),

$$\begin{aligned}\boldsymbol{\Sigma}^{(a)}(t) &= -\frac{1}{2}\sum_{b:b\neq a}\mathbf{F}^{(ab)}(t)(\mathbf{x}^{(a)}(t) - \mathbf{x}^{(b)}(t))^T \\ &= -\frac{1}{2}\sum_{b:b\neq a} f^{(ab)}(t)(\mathbf{x}^{(a)}(t) - \mathbf{x}^{(b)}(t))(\mathbf{x}^{(a)}(t) - \mathbf{x}^{(b)}(t))^T,\end{aligned}$$

to deduce that $\boldsymbol{\Sigma}^{(a)}$ is always symmetric, so is its approximation $\boldsymbol{\sigma}$. This reduces the number of unknowns to $3 + 1 + (3 + 2 + 1) = 10$.

2.2 Lagrangian dynamics

In the previous section, we derived two equations (continuity and Cauchy's momentum) for the evolution of mass and velocity by introducing the new variable stress tensor. Notice the equations themselves do not require any knowledge of the material's *"resting state"*, that is, a canonical state of the material where no external force is applied and the material is not moving. While the states in which the material is moving is often of interests (they are called *"equilibria"*), the equilibrium in which no external force is present is especially important in the analysis of elastoplastic materials. In this section and the sections after, we will consider a time domain $[0, \infty)$ instead of $\mathbb{R}$, and the material is at it's *resting configuration* at time $t = 0$.

Suppose the material consisting of the particles $\{P^{(a)}\}_{a=1}^N$ occupies the spatial domain

$\Omega_0 \subseteq \mathbb{R}^3$ at time 0. As particles travel along its trajectory $\mathbf{x}^{(a)}(t)$ with time, the material also changes its shape (and volume, too). For any given time t, we denote the spatial domain where the material occupies as $\Omega_t \subseteq \mathbb{R}^3$.

The major distinction between Eulerian dynamics and Lagrangian dynamics is the difference in the domain of the mathematical functions discussed. In Eulerian dynamics, for instance, the velocity of the material is approximated by a velocity field function $\mathbf{u} : \Omega_t \to \mathbb{R}^3$ in ways that

$$\mathbf{u}(t, \mathbf{x}^{(a)}(t)) \approx \dot{\mathbf{x}}^{(a)}(t).$$

In this section, we shall discuss a different mathematical function $\mathbf{V} : \Omega_0 \to \mathbb{R}^3$ that approximates the material velocity in ways that

$$\mathbf{V}(t, \mathbf{x}^{(a)}(0)) \approx \dot{\mathbf{x}}^{(a)}(t).$$

For those who savvy differential geometry, the velocity function $\mathbf{u} : \Omega_t \to T\Omega_t = \mathbb{R}^3$ here is really a section in the tangent bundle. The notion $T\Omega_t$ helps draw a distinction between the physical space Ω_t and the tangent space $T\Omega_t$. We borrow the language of differential geometry sometimes only to help clarify the derivation. One does not need to know much about differential geometry to proceed.

2.2.1 Flow map, deformation gradient, mass density, and velocity

Assume **there exists a flow map $\boldsymbol{\phi} : [0, \infty) \times \Omega^0 \to \mathbb{R}^3$ that represents the trajectories of the particles**, that is, provided time t and the particle initial position $\mathbf{x}^{(a)}(0)$,

$$\boldsymbol{\phi}(t, \mathbf{x}^{(a)}(0)) = \mathbf{x}^{(a)}(t). \tag{2.25}$$

We introduce new symbol $\mathbf{X}^{(a)}$ to denote the initial position $\mathbf{x}^{(a)}(0)$, and $\mathbf{x}^{(a)}$ to denote the deformed position $\mathbf{x}^{(a)}(t)$ when time t is implicitly specified. As such, we can write the flow

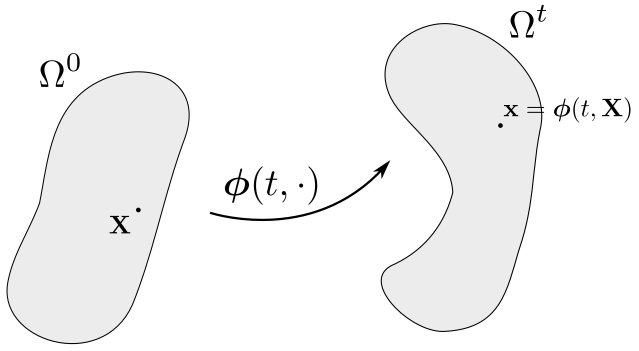
Ω^0
Ω^t
$\mathbf{x} = \boldsymbol{\phi}(t, \mathbf{X})$
$\boldsymbol{\phi}(t, \cdot)$
$\mathbf{X}$

Figure 2.1: **Flow map.**

map as

$$\boldsymbol{\phi}(t, \mathbf{X}) = \mathbf{x}. \tag{2.26}$$

We also introduce a symbol $\mathbf{F}$ for the spatial derivative of the flow map,

$$\mathbf{F}(t, \mathbf{X}) := \frac{\partial}{\partial \mathbf{X}} \boldsymbol{\phi}(t, \mathbf{X}). \tag{2.27}$$

This is often called the *deformation gradient*. Notice that it is implicitly assumed that **the flow map $\boldsymbol{\phi}$ is somewhat smooth** — at least smooth enough to permit spatial derivative $\frac{\partial}{\partial \mathbf{X}}\boldsymbol{\phi}$ and time derivatives, $\frac{\partial}{\partial t}\boldsymbol{\phi}$ and $\frac{\partial^2}{\partial t^2}\boldsymbol{\phi}$, as we will use in later discussion. The readers should be aware that the regularity of the solutions to non-linear elasticity or non-linear elastoplasticity remains an open problem. We don't cover the discussion of regularity in this dissertation.

The deformation gradient measures the local distortion of the material. It maps a tangent vector $d\mathbf{X} \in T\Omega_0$ to $d\mathbf{x} \in T\Omega_t$, or—for the differential geometry savvy readers—pulls k-form $\omega \in \Omega^k(\Omega_t)$ back to $\boldsymbol{\phi}_t^* \omega \in \Omega^k(\Omega_0)$. To provide more intuition, consider the singular value decomposition of $\mathbf{F}$,

$$\mathbf{F} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T. \tag{2.28}$$

If the material is undergoing rigid motion, that is, $\boldsymbol{\phi}(t, \mathbf{X}) = \mathbf{R}(t)(\mathbf{X} - \mathbf{X}_0) + \mathbf{d}(t)$ for some base point $\mathbf{X}_0 \in \Omega^0$, rotation $\mathbf{R}(t) \in SO(3)$ and translation $\mathbf{d}(t) = \boldsymbol{\phi}(t, \mathbf{X}_0) \in \mathbb{R}^3$, then

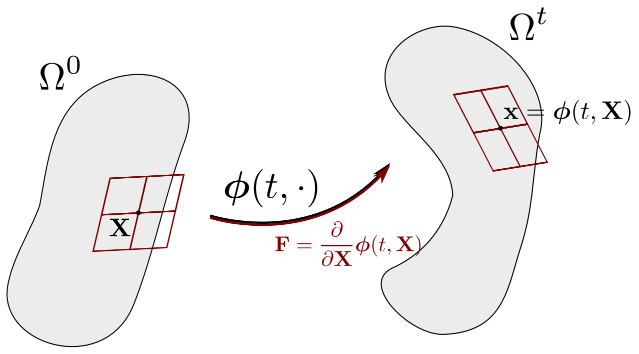
Ω^0
Ω^t
$\mathbf{x} = \phi(t, \mathbf{X})$
$\phi(t, \cdot)$
$\mathbf{X}$
$\mathbf{F} = \frac{\partial}{\partial \mathbf{X}} \phi(t, \mathbf{X})$

Figure 2.2: **Deformation gradient.**

$\mathbf{F}(t, \cdot) \equiv \mathbf{R}(t)$ is simply the rotation and $\mathbf{\Sigma} = \mathbf{I}$. On the other hand, if $\mathbf{\Sigma}$ has an entry smaller than 1, a local compression is taking place. Similarly, if $\mathbf{\Sigma}$ has an entry bigger than 1, a local expansion is taking place. The determinant of $\mathbf{F}$, denoted by $J(t, \mathbf{X}) := \det(\mathbf{F}(t, \mathbf{X}))$, is usually called the *Jacobian.* With the singular value decomposition,

$$\begin{aligned} J &= \det(\mathbf{F}) \\ &= \det(\mathbf{U}) \det(\mathbf{\Sigma}) \det(\mathbf{V}^T) \\ &= \det(\mathbf{\Sigma}) \\ &= \prod_{i=1}^{3} \sigma_i. \end{aligned}$$

J is the product of the singular values of $\mathbf{F}$, and amounts to the rate of local volume change induced by the flow map $\boldsymbol{\phi}(t, \cdot)$.

As articulated in (2.12), the Eulerian mass density $\rho(t, \mathbf{x})$ was characterized by integrating with test function $\psi \in C_c^\infty(\mathbb{R} \times \mathbb{R}^3)$. The Lagrangian mass density function should be defined in similar fashion. The discrete measure of the Lagrangian mass distribution is given by

$$\mu^L_{mass} = \sum_{a=1}^{N} m^{(a)} \delta_{\mathbf{X}^{(a)}}. \tag{2.29}$$

Note that unlike its Eulerian counterpart, this mass measure does not depend on time t, and is used solely in Lagrangian coordinates Ω_0. We assume **there exists a function** $R : \Omega_0 \to \mathbb{R}^+$

that approximates μ^L_{mass}, that is,

$$\mu^L_{mass} \approx R(\mathbf{X})d\mathcal{L}(\mathbf{X}). \tag{2.30}$$

This approximation is similarly articulated by testing with smooth, compactly-supported functions,

$$\forall \phi \in C^\infty_c(\Omega_0), \sum_{a=1}^N m^{(a)}\phi(\mathbf{X}^{(a)}) \approx \int_{\Omega_0} R(\mathbf{X})\phi(\mathbf{X})d\mathbf{X}. \tag{2.31}$$

This approximation itself is enough to provide some interesting insights. The Eulerian mass density works by integrating $\psi \in C^\infty_c([0,\infty)\times\mathbb{R}^3)$ on both space and time domain. However, take a sequence of nascent delta function $\eta_k \to \delta_t$ such that

$$\lim_{k\to\infty}\int_0^\infty \eta_k(s)\psi(s,\mathbf{x})ds = \psi(t,\mathbf{x}). \tag{2.32}$$

Since $\psi_k := \eta_k\psi \in C^\infty_c([0,\infty)\times\mathbb{R}^3)$, we have

$$\int_0^\infty\int_{\Omega_s} \eta_k(s)\psi(s,\mathbf{x})\rho(s,\mathbf{x})d\mathbf{x}ds \approx \int_0^\infty \eta_k(s)\sum_{a=1}^N m^{(a)}\psi(s,\mathbf{x}^{(a)}(s))ds$$

Let $k\to\infty$, we get

$$\int_{\Omega_t} \psi(t,\mathbf{x})\rho(t,\mathbf{x})d\mathbf{x}dt \approx \sum_{a=1}^N m^{(a)}\psi(t,\mathbf{x}^{(a)}(t)). \tag{2.33}$$

This shows that the mass measure approximation works on single time slice Ω_t as well. Now since $\boldsymbol{\phi}(t,\cdot)$ is a homeomorphism between Ω_0 and Ω_t, we consider the change of variable formula using the Jacobian $J = \det(\mathbf{F})$,

$$\begin{aligned}
\int_{\Omega_0} \psi(t,\boldsymbol{\phi}(t,\mathbf{X}))R(\mathbf{X})d\mathbf{X} &\approx \sum_{a=1}^N m^{(a)}\psi(t,\boldsymbol{\phi}(t,\mathbf{X}^{(a)})) \\
&= \sum_{a=1}^N m^{(a)}\psi(t,\mathbf{x}^{(a)}(t))) \\
&\approx \int_{\Omega_t} \psi(t,\mathbf{x})\rho(t,\mathbf{x})d\mathbf{x} \\
&= \int_{\Omega_0} \psi(t,\boldsymbol{\phi}(t,\mathbf{X}))\rho(t,\boldsymbol{\phi}(t,\mathbf{X}))\det\left(\frac{\partial\boldsymbol{\phi}}{\partial\mathbf{X}}(t,\mathbf{X})\right)d\mathbf{X} \\
&= \int_{\Omega_0} \psi(t,\boldsymbol{\phi}(t,\mathbf{X}))\rho(t,\boldsymbol{\phi}(t,\mathbf{X}))J(t,\mathbf{X})d\mathbf{X}.
\end{aligned}$$

Since ψ is arbitrary, the above equation is sometimes interpreted as

$$R(\mathbf{X}) = \rho(t, \boldsymbol{\phi}(t, \mathbf{X}))J(t, \mathbf{X}). \tag{2.34}$$

For those differential geometry savvy readers, this is a results due to that $\rho d\mathcal{L} \in \Omega^3(\Omega_t)$ and $Rd\mathcal{L} = \boldsymbol{\phi}_t^*(\rho d\mathcal{L}) \in \Omega^3(\Omega_0)$. ($\Omega_0$ and Ω_t are 3-dimensional manifolds.) The Jacobian arises naturally due to the pullback.

As one might wonder, in section 2.1 we assumed the existence of two functions, mass density ρ and velocity $\mathbf{u}$. What is velocity in Lagrangian view? We use the flow map $\boldsymbol{\phi}$ approximation (2.25) and take time derivative,

$$\frac{\partial}{\partial t}\boldsymbol{\phi}(t, \mathbf{X}^{(a)}) = \dot{\mathbf{x}}^{(a)}(t) \tag{2.35}$$

The momentum approximation (2.14) then becomes

$$\rho(t, \mathbf{x})\mathbf{u}(t, \mathbf{x})d\mathcal{L}(\mathbf{x}) \approx \sum_{a=1}^{N} m^{(a)}\dot{\mathbf{x}}^{(a)}(t)\delta_{\mathbf{x}^{(a)}(t)} = \sum_{a=1}^{N} m^{(a)}\frac{\partial \boldsymbol{\phi}}{\partial t}(t, \mathbf{X}^{(a)})\delta_{\boldsymbol{\phi}(t,\mathbf{X}^{(a)})}. \tag{2.36}$$

And for $\psi \in C_c^\infty([0, \infty) \times \mathbb{R}^3)$,

$$\begin{aligned}\int_0^\infty \int_{\Omega_t} \psi(t, \mathbf{x})\rho(t, \mathbf{x})\mathbf{u}(t, \mathbf{x})d\mathbf{x}dt &= \int_0^\infty \sum_{a=1}^{N} m^{(a)}\frac{\partial \boldsymbol{\phi}}{\partial t}(t, \mathbf{X}^{(a)})\psi(t, \boldsymbol{\phi}(t, \mathbf{X}^{(a)}))dt \quad \text{(by (2.15))}\\ &\approx \int_0^\infty \int_{\Omega_0} R(\mathbf{X})\frac{\partial \boldsymbol{\phi}}{\partial t}(t, \mathbf{X})\psi(t, \boldsymbol{\phi}(t, \mathbf{X}))d\mathbf{X}dt \quad \text{(by (2.31))}\end{aligned}$$

By changing variable,

$$\int_0^\infty \int_{\Omega_t} \psi(t, \mathbf{x})\rho(t, \mathbf{x})\mathbf{u}(t, \mathbf{x})d\mathbf{x}dt = \int_0^\infty \int_{\Omega_0} \psi(t, \boldsymbol{\phi}(t, \mathbf{X}))\rho(t, \boldsymbol{\phi}(t, \mathbf{X}))\mathbf{u}(t, \boldsymbol{\phi}(t, \mathbf{X}))J(t, \mathbf{X})d\mathbf{X}dt,$$

and letting ψ vary, we see that

$$R(\mathbf{X})\frac{\partial \boldsymbol{\phi}}{\partial t}(t, \mathbf{X}) = \rho(t, \boldsymbol{\phi}(t, \mathbf{X}))\mathbf{u}(t, \boldsymbol{\phi}(t, \mathbf{X}))J(t, \mathbf{X}).$$

Combining this with (2.34), we finally arrive at this seemingly trivial fact,

$$\frac{\partial \boldsymbol{\phi}}{\partial t}(t, \mathbf{X}) = \mathbf{u}(t, \boldsymbol{\phi}(t, \mathbf{X})). \tag{2.37}$$

We sometimes denote $\mathbf{V}(t, \mathbf{X}) := \frac{\partial \phi}{\partial t}(t, \mathbf{X})$ for simplicity. The reason for an extensive proof for $\frac{\partial \phi}{\partial t} = \boldsymbol{\phi}_t^* \mathbf{u}$ is the same as that for $R d\mathcal{L} = \boldsymbol{\phi}_t^*(\rho d\mathcal{L})$. Without careful arguments (or unmatched expertise in differential geometry), we can't make sure if the pullback is just a function composition, or a Jacobian should emerge. One will see in later sections that more complicated terms could emerge from pullbacks.

The absence of continuity equation

Note the continuity equation arises from the fact that $\mu_{mass}(t) \approx \rho(t, \cdot) d\mathcal{L}$ is time-dependent. In the Lagrangian view, the mass density $R d\mathcal{L}$ is not time-dependent. One can think of the continuity equation as intrinsically implied by the way Lagrangian functions are defined. The same is however not true for Cauchy's momentum equation. In section 2.2.3, we will see an equivalent version of the Cauchy's momentum equation in Lagrangian coordinates.

2.2.2 Tensors

In the discussion concerning continuum dynamics, we often focus on domains that are simply subsets of $\mathbb{R}^3$. In such a setting, the tangent space is isomorphic to $\mathbb{R}^3$ itself, and many structural distinctions between a tangent vector and a point are lost. On top of it, the canonical basis $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ of $\mathbb{R}^3$ makes the change between Lagrangian and Eulerian coordinates rather confusing. In this section, we set up a framework to analyze relationships between Eulerian and Lagrangian functions.

Viewing our domain Ω as a 3-dimensional manifold (a topological space that is *locally* homeomorphic to open subsets of $\mathbb{R}^3$). The tangent space of a point $p \in \Omega$, denoted as $T_p\Omega$, is the collection of infinitesimal *curves* in Ω passing through p. At the infinitesimal scale, each curve boils down to its tangent vector at p. The collection of tangent spaces of all points in

Ω is denoted as

$$T\Omega := \coprod_{p\in\Omega} T_p\Omega \tag{2.38}$$

A *vector field* is a function $\mathbf{f}$ that associates a tangent vector $\mathbf{f}(p) \in T_p\Omega$ for each point $p \in \Omega$. We can write it as

$$\begin{aligned} \mathbf{f} &: \Omega \to T\Omega, \\ \mathbf{f}(p) &\in T_p\Omega. \end{aligned} \tag{2.39}$$

Or sometimes we denote it as $\mathbf{f} \in \Gamma(T\Omega)$. One example of a vector filed is the velocity field $\mathbf{u} : \Omega_t \to T\Omega_t$. On the other hand, from the definition $\mathbf{V}(t, \mathbf{X}) = \frac{\partial \boldsymbol{\phi}}{\partial t}(t, \mathbf{X})$, we can see that each value $\mathbf{V}$ takes is the derivative of a curve moving in $\mathbb{R}^3$, an element of $T_{\boldsymbol{\phi}(t,\mathbf{X})}\Omega_t$. The relation between an Eulerian function (e.g. $\mathbf{u}$) and its Lagrangian counterpart (e.g. $\mathbf{V}$) is called *pullback*. Consider the flow map $\boldsymbol{\phi}(t, \cdot) =: \boldsymbol{\phi}_t$ as a one-parameter family of maps between manifolds $\Omega_0 \to \Omega_t$. For a fixed time t, the Lagrangian velocity $\mathbf{V}_t = \mathbf{u}_t \circ \boldsymbol{\phi}_t$ is the pullback of the Eulerian velocity $\mathbf{u}_t$.

In later discussion, it benefits to consider *push-forward* as well, the reverse of pullback. Denote the inverse of $\boldsymbol{\phi}_t$ (as a one-parameter family of maps)

$$\Upsilon_t := \boldsymbol{\phi}_t^{-1}. \tag{2.40}$$

This map satisfies

$$\Upsilon_t(\boldsymbol{\phi}_t(\mathbf{X})) = \mathbf{X}. \tag{2.41}$$

Take partial derivative with regard to $\mathbf{X}$, we get

$$\begin{aligned} \frac{\partial \Upsilon}{\partial \mathbf{x}}(t, \boldsymbol{\phi}_t(\mathbf{X})) \cdot \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}}(t, \mathbf{X}) &= \mathbf{I}, \\ \frac{\partial \Upsilon_t}{\partial \mathbf{x}} \circ \boldsymbol{\phi}_t &= \left(\frac{\partial \boldsymbol{\phi}_t}{\partial \mathbf{X}}\right)^{-1}. \end{aligned}$$

Take partial derivative with regard to t, we get

$$\frac{\partial \Upsilon}{\partial t}(t, \boldsymbol{\phi}_t(\mathbf{X})) + \frac{\partial \Upsilon}{\partial \mathbf{x}}(t, \boldsymbol{\phi}_t(\mathbf{X})) \cdot \frac{\partial \boldsymbol{\phi}_t}{\partial t}(t, \mathbf{X}) = 0,$$

$$\frac{\partial \Upsilon_t}{\partial t} \circ \boldsymbol{\phi}_t = -\left(\frac{\partial \boldsymbol{\phi}_t}{\partial \mathbf{X}}\right)^{-1} \frac{\partial \boldsymbol{\phi}_t}{\partial t} = -\mathbf{F}^{-1}\mathbf{V}.$$

We will use these identities in the derivations to come.

2.2.3 Momentum and constitutive equation

The constitutive equation describes the relationship between stress and other state variables of the material (e.g., velocity, velocity gradient, position, deformation gradient, etc.). In this dissertation, we focus on the type of materials that are *hyperelastic*. Here is the definition of elasticity.

Elasticity

The stress of an *elastic* material only depends on the current deformation gradient of the same spot.

$$\boldsymbol{\sigma}(t, \boldsymbol{\phi}(t, \mathbf{X})) = \boldsymbol{\sigma}(\mathbf{F}(t, \mathbf{X})). \tag{2.42}$$

Hyperelasticity is a subclass of elasticity such that the work done by the stresses during a deformation process is dependent only on the initial state at time t_0 and the final configuration at time t. We also call such material *path-independent*. See [BW08] for more motivation and theories in mechanics. Materials of this type satisfy the *least action principle*, that is, the material flow $\boldsymbol{\phi}$ of time interval $[0, T]$ satisfies

$$\boldsymbol{\phi} = \operatorname{argmin} \int_0^T \int_{\Omega_0} \frac{1}{2} R(\mathbf{X}) \left| \frac{\partial \boldsymbol{\phi}}{\partial t}(t, \mathbf{X}) \right|^2 - \psi\left(\frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}}(t, \mathbf{X})\right) d\mathbf{X} dt. \tag{2.43}$$

The integrand $\frac{1}{2}R|\mathbf{V}|^2 - \psi(\mathbf{F})$ is often called the *non-relativistic Lagrangian* of the system. Here $\psi(\mathbf{F})$ is the *elastic potential energy density*. It is a material-dependent, and usually a norm-like function. The integral of this energy density,

$$\Psi[\boldsymbol{\phi}] = \int_0^T \int_{\Omega_0} \psi\left(\frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}}(t, \mathbf{X})\right) d\mathbf{X} dt, \tag{2.44}$$

amounts to the total work done by the elasticity of the material from the initial to the current position. Meanwhile, the other integral

$$\mathcal{K}[\boldsymbol{\phi}] = \int_0^T \int_{\Omega_0} \frac{1}{2} R \|\mathbf{V}|^2 d\mathbf{X} dt, \tag{2.45}$$

is called the *total kinetic energy* of the system.

Denote the objective of the minimization as

$$\mathcal{S}[\boldsymbol{\phi}] := \int_0^T \int_{\Omega_0} \frac{1}{2} R(t, \mathbf{X}) \left| \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}}(t, \mathbf{X}) \right|^2 - \psi \left(\frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}}(t, \mathbf{X}) \right) d\mathbf{X} dt. \tag{2.46}$$

Taking a variation with $\boldsymbol{\phi}^\epsilon := \boldsymbol{\phi} + \epsilon \widetilde{\boldsymbol{\phi}}$,

$$\begin{aligned} \frac{d}{d\epsilon} \mathcal{S}[\boldsymbol{\phi}^\epsilon; 0, T] \bigg|_{\epsilon=0} &= \int_0^T \int_{\Omega_0} R \frac{\partial \boldsymbol{\phi}}{\partial t} \cdot \frac{\partial \widetilde{\boldsymbol{\phi}}}{\partial t} - \frac{\partial \psi}{\partial \mathbf{F}} \left(\frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}} \right) : \frac{\partial \widetilde{\boldsymbol{\phi}}}{\partial \mathbf{X}} d\mathbf{X} dt \qquad (2.47) \\ &= \int_0^T \int_{\Omega_0} -R \frac{\partial^2 \boldsymbol{\phi}}{\partial t^2} \cdot \widetilde{\boldsymbol{\phi}} + \nabla_{\mathbf{X}} \cdot \left(\frac{\partial \psi}{\partial \mathbf{F}} \left(\frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}} \right) \right) \cdot \widetilde{\boldsymbol{\phi}} d\mathbf{X} dt \\ &= \int_0^T \int_{\Omega_0} \left(-R \frac{\partial^2 \boldsymbol{\phi}}{\partial t^2} + \nabla_{\mathbf{X}} \cdot \left(\frac{\partial \psi}{\partial \mathbf{F}} \left(\frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}} \right) \right) \right) \cdot \widetilde{\boldsymbol{\phi}} d\mathbf{X} dt. \end{aligned}$$

Since $\widetilde{\boldsymbol{\phi}}$ is arbitrary, we know the minimizer $\boldsymbol{\phi}$ for (2.43) must satisfy the first order optimality condition,

$$R \frac{\partial^2 \boldsymbol{\phi}}{\partial t^2} - \nabla_{\mathbf{X}} \cdot \left(\frac{\partial \psi}{\partial \mathbf{F}} \left(\frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}} \right) \right) = 0. \tag{2.48}$$

This is the *momentum equation* in Lagrangian view, as we shall see. If one composite this equation with the inverse flow map $\Upsilon : \Omega_t \to \Omega_0$, the left-hand side becomes

$$\left(R \frac{\partial^2 \boldsymbol{\phi}}{\partial t^2} \right) \circ \Upsilon = \rho (J \circ \Upsilon) D_t \mathbf{u}.$$

This is almost the left hand side of the Cauchy's momentum equation (2.24) except differing by a push-forward Jacobian $J \circ \Upsilon$. Define the *first Piola-Kirchhoff stress* $\mathbf{P}$,

$$\mathbf{P}(t, \mathbf{X}) := \frac{\partial \psi}{\partial \mathbf{F}} (\mathbf{F}(t, \mathbf{X})). \tag{2.49}$$

Consider the push-forward $\tilde{\mathbf{f}} := \widetilde{\boldsymbol{\phi}} \circ \Upsilon$ of the perturbation $\widetilde{\boldsymbol{\phi}}$, or equivalently,

$$\widetilde{\boldsymbol{\phi}} = \tilde{\mathbf{f}} \circ \boldsymbol{\phi}.$$

By chain rule,

$$\frac{\partial \widetilde{\boldsymbol{\phi}}}{\partial \mathbf{X}} = \left(\frac{\partial \tilde{\mathbf{f}}}{\partial \mathbf{x}} \circ \boldsymbol{\phi}\right) \cdot \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}}.$$

We perform change of variables to (2.47), noting $d\mathbf{X} = (J \circ \Upsilon)^{-1} d\mathbf{x}$,

$$\begin{aligned} 0 &= \int_0^T \int_{\Omega_0} R \frac{\partial \boldsymbol{\phi}}{\partial t} \cdot \frac{\partial \widetilde{\boldsymbol{\phi}}}{\partial t} - \mathbf{P} : \frac{\partial \widetilde{\boldsymbol{\phi}}}{\partial \mathbf{X}} d\mathbf{X} dt \\ &= \int_0^T \int_{\Omega_t} \left(\rho (J \circ \Upsilon) \mathbf{u} \cdot D_t \tilde{\mathbf{f}} - (\mathbf{P} \circ \Upsilon) : \left(\frac{\partial \tilde{\mathbf{f}}}{\partial \mathbf{x}} \cdot \left(\frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}} \circ \Upsilon \right) \right) \right) (J \circ \Upsilon)^{-1} d\mathbf{x} dt. \qquad (2.50) \end{aligned}$$

The second term is made easier if one looks at matrix inner product as trace, and use its cyclic rule,

$$\mathbf{A} : \mathbf{B} = \operatorname{tr}(\mathbf{A}^T \mathbf{B}); \quad \operatorname{tr}(\mathbf{ABC}) = \operatorname{tr}(\mathbf{CAB})$$

$$\begin{aligned} \frac{1}{J \circ \Upsilon} (\mathbf{P} \circ \Upsilon) : \left(\frac{\partial \tilde{\mathbf{f}}}{\partial \mathbf{x}} \cdot \left(\frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}} \circ \Upsilon \right) \right) &= \frac{1}{J \circ \Upsilon} \operatorname{tr} \left((\mathbf{P} \circ \Upsilon)^T \cdot \frac{\partial \tilde{\mathbf{f}}}{\partial \mathbf{x}} \cdot \left(\frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}} \circ \Upsilon \right) \right) \\ &= \frac{1}{J \circ \Upsilon} \operatorname{tr} \left(\left(\frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}} \circ \Upsilon \right) \cdot (\mathbf{P} \circ \Upsilon)^T \cdot \frac{\partial \tilde{\mathbf{f}}}{\partial \mathbf{x}} \right) \\ &= \frac{1}{J \circ \Upsilon} \left((\mathbf{P} \circ \Upsilon) \left(\frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}} \circ \Upsilon \right)^T \right) : \frac{\partial \tilde{\mathbf{f}}}{\partial \mathbf{x}} \\ &= \left(\left(\frac{1}{J} \mathbf{P} \mathbf{F}^T \right) \circ \Upsilon \right) : \frac{\partial \tilde{\mathbf{f}}}{\partial \mathbf{x}}. \end{aligned}$$

Plug this back to (2.50),

$$\begin{aligned} 0 &= \int_0^T \int_{\Omega_t} \rho \mathbf{u} \cdot D_t \tilde{\mathbf{f}} - \left(\left(\frac{1}{J} \mathbf{P} \mathbf{F}^T \right) \circ \Upsilon \right) : \frac{\partial \tilde{\mathbf{f}}}{\partial \mathbf{x}} d\mathbf{x} dt \\ &= \int_0^T \int_{\Omega_t} -D_t(\rho \mathbf{u}) \cdot \tilde{\mathbf{f}} + \nabla_{\mathbf{x}} \cdot \left(\left(\frac{1}{J} \mathbf{P} \mathbf{F}^T \right) \circ \Upsilon \right) \cdot \tilde{\mathbf{f}} d\mathbf{x} dt \\ &= \int_0^T \int_{\Omega_t} -\rho D_t(\mathbf{u}) \cdot \tilde{\mathbf{f}} + \nabla_{\mathbf{x}} \cdot \left(\left(\frac{1}{J} \mathbf{P} \mathbf{F}^T \right) \circ \Upsilon \right) \cdot \tilde{\mathbf{f}} d\mathbf{x} dt. \qquad (D_t \rho = 0 \text{ by } (2.18)) \end{aligned}$$

Since $\widetilde{\boldsymbol{\phi}}$ is arbitrary, so is $\tilde{\mathbf{f}} = \widetilde{\boldsymbol{\phi}} \circ \Upsilon$. We circled back to the Eulerian momentum equation,

$$\rho D_t(\mathbf{u}) - \nabla_{\mathbf{x}} \left(\left(\frac{1}{J} \mathbf{P} \mathbf{F}^T \right) \circ \Upsilon \right) = 0. \qquad (2.51)$$

We hence deduce the relation between the Cauchy's stress $\boldsymbol{\sigma}$ and the first Piola Kirchhoff stress $\mathbf{P}$,

$$\boldsymbol{\sigma}(t,\mathbf{x}) = \left(\frac{1}{J}\mathbf{P}\mathbf{F}^T\right)(t,\Upsilon(t,\mathbf{x})), \tag{2.52}$$

or,

$$\boldsymbol{\sigma}(t,\boldsymbol{\phi}(t,\mathbf{X})) = \left(\frac{1}{J}\mathbf{P}\mathbf{F}^T\right)(t,\mathbf{X}) = \left(\frac{\partial\psi}{\partial\mathbf{F}}(\mathbf{F})\mathrm{cof}(\mathbf{F})^{-1}\right)(t,\mathbf{X}). \tag{2.53}$$

Remark

Near the end of Section 2.1, we had $1+3$ equations and $3+1+(3+2+1)$ unknowns. In the Lagrangian view, once provided the elastic potential energy density ψ, we have $1+3$ equations, and exactly $1+3$ unknowns (the stress tensor as a variable is entirely determined by $\mathbf{F}$ using $\frac{\partial\psi}{\partial\mathbf{F}}$). In Section 2.3, we shall see that ψ can depend on additional variables for plasticity, and it would affect $\boldsymbol{\phi}, \mathbf{V}$ through the Lagrangian momentum equation (2.48).

2.3 Plasticity

According to hyperelastic rule, since the elastic potential energy density is a norm-like function, its derivative $\mathbf{P} = \frac{\partial\psi}{\partial\mathbf{F}}(\mathbf{F})$ is monotonically increasing function in $\mathbf{F}$. As a result, the stress can go to infinity (in norm) if the deformation gradient goes to infinity. However, most material cannot exert infinite stress. Most materials, especially the ones prone to fracture, undergoes a *plastic* phase where the stress stops corresponding to the increasing deformation when it exceeds a certain threshold.

To analyze this behavior, we first introduce the *multiplicative decomposition* of deformation gradient,

$$\mathbf{F} = \mathbf{F}_E\mathbf{F}_P. \tag{2.54}$$

The *plastic deformation gradient* $\mathbf{F}^P$ is an internal variable that records the permanent deformation due to the excess in stress. While the additive type of models is common in

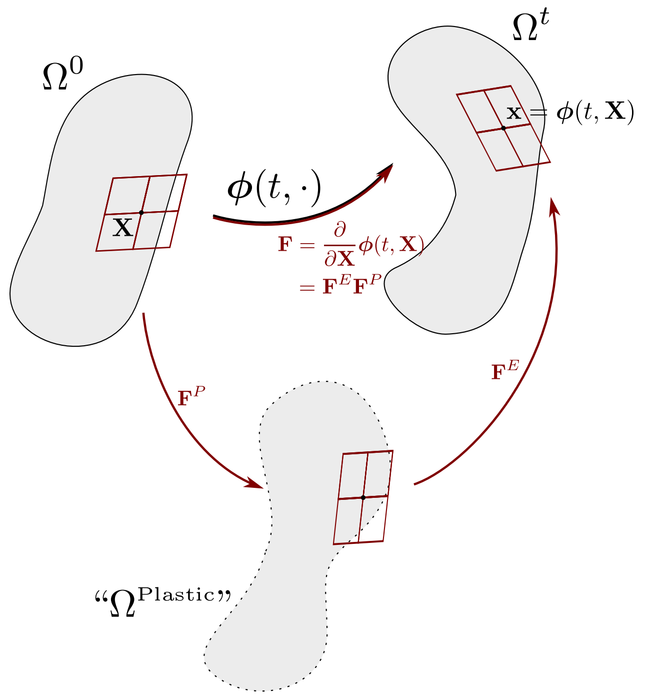
Ω^0
Ω^t
$\mathbf{x} = \phi(t, \mathbf{X})$
$\phi(t, \cdot)$
$\mathbf{X}$
$\mathbf{F} = \frac{\partial}{\partial \mathbf{X}}\phi(t, \mathbf{X})$
$= \mathbf{F}^E\mathbf{F}^P$
$\mathbf{F}^P$
$\mathbf{F}^E$
"Ω^{Plastic}"

Figure 2.3: **Plastic deformation.**

engineering, it is only accurate up to small deformation. In graphics, we often concern simulation in which the material undergoes massive deformation and rotation. We chose multiplicative models as the theoretical foundation to accurately capture the deformation- and rotation-full simulations presented in this dissertation. See Figure 2.3.

The elastic potential energy density now depends on both $\mathbf{F}_E$ and $\mathbf{F}_P$, written as

$$\psi(\mathbf{F}_E, \mathbf{F}_P). \tag{2.55}$$

The stress tensor is then given by

$$\begin{aligned}\mathbf{P} = \frac{\partial \psi}{\partial \mathbf{F}} &= \frac{\partial \psi}{\partial \mathbf{F}_E} : \frac{\partial \mathbf{F}_E}{\partial \mathbf{F}} + \frac{\partial \psi}{\partial \mathbf{F}_P} : \frac{\partial \mathbf{F}_P}{\partial \mathbf{F}} \\ &= \frac{\partial \psi}{\partial \mathbf{F}_E}\mathbf{F}_P^{-T} + \mathbf{F}_E^{-T}\frac{\partial \psi}{\partial \mathbf{F}_P}.\end{aligned} \tag{2.56}$$

In the discussion of plasticity, we use *Kirchhoff stress* to describe the stress threshold. It is defined by

$$\boldsymbol{\tau} = J\boldsymbol{\sigma} = \mathbf{P}\mathbf{F}^T. \tag{2.57}$$

The constraint on stress is then given by

$$f(\boldsymbol{\tau}) \leq 0. \tag{2.58}$$

The hypersurface $\{\boldsymbol{\tau} : f(\boldsymbol{\tau}) = 0\}$ is called *yield surface*, and we can rewrite the Lagrangian minimization problem from least action principle,

$$\begin{aligned} \text{minimize} \quad & \int_0^T \int_{\Omega_0} \frac{1}{2} R|\mathbf{V}|^2 - \psi\left(\mathbf{F}_E, \mathbf{F}_P\right) d\mathbf{X} dt, \\ \text{subject to} \quad & f(\boldsymbol{\tau}) \leq 0. \end{aligned} \tag{2.59}$$

At time $t = 0$, the value of $\mathbf{F}_P$ is identity everywhere, representing a neutral initial state. During the deformation process, the value $\mathbf{F}_P$ changes whenever the stress reaches the yield surface. It acts as the Lagrangian multiplier to "absorb" the excessive deformation, keeping $\mathbf{F}_E$ in the region such that the stress $\boldsymbol{\tau}$ remains within the yield surface.

2.3.1 Rates of plastic flow

The rate of change of the plastic decomposition $\mathbf{F} = \mathbf{F}_E\mathbf{F}_P$ is

$$\begin{aligned} \dot{\mathbf{F}} &= \dot{\mathbf{F}}_E\mathbf{F}_P + \mathbf{F}_E\dot{\mathbf{F}}_P \\ \dot{\mathbf{F}}_E &= \dot{\mathbf{F}}\mathbf{F}_P^{-1} - \mathbf{F}_E\dot{\mathbf{F}}_P\mathbf{F}_P^{-1} \\ \dot{\mathbf{F}}_P &= {\mathbf{F}_E}^{-1}\dot{\mathbf{F}} - {\mathbf{F}_E}^{-1}\dot{\mathbf{F}}_E\mathbf{F}_P. \end{aligned} \tag{2.60}$$

Furthermore, defining $\mathbf{b}_E = \mathbf{F}_E\mathbf{F}_E^T$ as the elastic right Cauchy-Green strain and using Equations (2.60), we can see that

$$\dot{\mathbf{b}}_E = \frac{\partial \mathbf{v}}{\partial \mathbf{x}}\mathbf{b}_E + \mathbf{b}_E\frac{\partial \mathbf{v}}{\partial \mathbf{x}}^T + \mathcal{L}_{\mathbf{v}}\mathbf{b}_E, \tag{2.61}$$

where

$$\mathcal{L}_{\mathbf{v}}\mathbf{b}_E = \mathbf{F}\dot{\mathbf{C}}_p^{-1}\mathbf{F}^T = -\mathbf{F}\mathbf{C}_p^{-1}\dot{\mathbf{C}}_p\mathbf{C}_p^{-1}\mathbf{F}^T = -\mathbf{F}\mathbf{C}_p^{-1}\dot{\mathbf{F}}_P^T\mathbf{F}_P\mathbf{C}_p^{-1}\mathbf{F}^T - \mathbf{F}\mathbf{C}_p^{-1}\mathbf{F}_P^T\dot{\mathbf{F}}_P\mathbf{C}_p^{-1}\mathbf{F}^T, \tag{2.62}$$

and $\mathbf{C}_p = \mathbf{F}_P^T\mathbf{F}_P$ is the plastic left Cauchy-Green strain. It is convenient to use the notation

$$\dot{\mathbf{b}}_E = \dot{\mathbf{b}}_E|_{\dot{\mathbf{F}}_P=\mathbf{0}} + \mathcal{L}_{\mathbf{v}}\mathbf{b}_E,\ \dot{\mathbf{b}}_E|_{\dot{\mathbf{F}}_P=\mathbf{0}} = \frac{\partial \mathbf{v}}{\partial \mathbf{x}}\mathbf{b}_E + \mathbf{b}_E\frac{\partial \mathbf{v}}{\partial \mathbf{x}}^T. \tag{2.63}$$

The elastic Hencky strain $\boldsymbol{\epsilon}_E$ is defined as

$$\boldsymbol{\epsilon}_E = \frac{1}{2}\log\left(\mathbf{b}_E\right). \tag{2.64}$$

See Section 2.3.4 for more details on functions defined this way. The rate of change of the elastic Hencky strain is given by

$$\dot{\boldsymbol{\epsilon}}_E = \left([\mathbf{B}](\mathbf{b}_E)\circ[\dot{\mathbf{b}}_E]\right)_{kl}\mathbf{u}_k\otimes\mathbf{u}_l \tag{2.65}$$

where $\mathbf{b}_E = \sum_i {\lambda_i^E}^2\mathbf{u}_i\otimes\mathbf{u}_i$, $\boldsymbol{\epsilon}_E = \sum_i \log\left(\lambda_i^E\right)\mathbf{u}_i\otimes\mathbf{u}_i$, $[\dot{\mathbf{b}}_E]_{ij} = \mathbf{u}_i\cdot\left(\dot{\mathbf{b}}_E\mathbf{u}_j\right)$ are the components of $\dot{\mathbf{b}}_E$ in the eigen basis and

$$[\mathbf{B}](\mathbf{b}_E) = \begin{pmatrix} \frac{1}{2\lambda_{E1}^2} & \frac{\log(\lambda_{E1})-\log(\lambda_{E2})}{\lambda_{E1}^2-\lambda_{E2}^2} & \frac{\log(\lambda_{E1})-\log(\lambda_{E3})}{\lambda_{E1}^2-\lambda_{E3}^2} \\ \frac{\log(\lambda_{E2})-\log(\lambda_{E1})}{\lambda_{E2}^2-\lambda_{E1}^2} & \frac{1}{2\lambda_{E2}^2} & \frac{\log(\lambda_{E2})-\log(\lambda_{E3})}{\lambda_{E2}^2-\lambda_{E3}^2} \\ \frac{\log(\lambda_{E3})-\log(\lambda_{E1})}{\lambda_{E3}^2-\lambda_{E1}^2} & \frac{\log(\lambda_{E3})-\log(\lambda_{E2})}{\lambda_{E3}^2-\lambda_{E2}^2} & \frac{1}{2\lambda_{E3}^2} \end{pmatrix}. \tag{2.66}$$

See Section 2.3.4 and Equation (2.94) for the derivation.

2.3.1.1 Energy dissipation and stress/rate pairs

The energy at time t of the material in $B^0 \subseteq \Omega_0$ is

$$E(t;B^0) = \int_{B^0}\frac{R(\mathbf{X},0)}{2}\left|\mathbf{V}(\mathbf{X},t)\right|_2^2 d\mathbf{X} + \int_{B^0}\psi(\mathbf{F}_E(\mathbf{X},t),\mathbf{F}_P(\mathbf{X},t))d\mathbf{X}. \tag{2.67}$$

with $\mathbf{P} = \frac{\partial\psi}{\partial\mathbf{F}_E}(\mathbf{F}_E,\mathbf{F}_P)\mathbf{F}_P^{-T}$. To avoid confusion when changing variables, we denote $R(\mathbf{X}) = R(\mathbf{X},0)$ in this session to emphasize its Lagrangian nature. The rate of change of the energy is

$$\frac{d}{dt}E(t;B^0) = \int_{B^0} R(\mathbf{X},0)\mathbf{V}(\mathbf{X},t)\mathbf{A}(\mathbf{X},t)d\mathbf{X} + \int_{B^0} \frac{\partial\psi}{\partial\mathbf{F}_E}(\mathbf{F}_E(\mathbf{X},t),\mathbf{F}_P(\mathbf{X},t)) : \dot{\mathbf{F}}_E(\mathbf{X},t)d\mathbf{X}$$
$$+ \int_{B^0} \frac{\partial\psi}{\partial\mathbf{F}_P}(\mathbf{F}_E(\mathbf{X},t),\mathbf{F}_P(\mathbf{X},t)) : \dot{\mathbf{F}}_P(\mathbf{X},t)d\mathbf{X}. \tag{2.68}$$

The second term can be reduced to

$$\begin{aligned}\int_{B^0} \frac{\partial\psi}{\partial\mathbf{F}_E}(\mathbf{F}_E,\mathbf{F}_P) : \dot{\mathbf{F}}_E d\mathbf{X} &= \int_{B^0} \frac{\partial\psi}{\partial\mathbf{F}_E}(\mathbf{F}_E,\mathbf{F}_P) : \left(\dot{\mathbf{F}}\mathbf{F}_P^{-1} - \mathbf{F}_E\dot{\mathbf{F}}_P\mathbf{F}_P^{-1}\right) d\mathbf{X} \\ &= \int_{B^0} \left(\frac{\partial\psi}{\partial\mathbf{F}_E}(\mathbf{F}_E,\mathbf{F}_P)\mathbf{F}_P^{-T}\right) : \dot{\mathbf{F}} - \left(\mathbf{F}_E^T\frac{\partial\psi}{\partial\mathbf{F}_E}(\mathbf{F}_E,\mathbf{F}_P)\mathbf{F}_P^{-T}\right) : (\dot{\mathbf{F}}_P)d\mathbf{X} \\ &= \int_{B^0} \mathbf{P} : \frac{\partial\mathbf{V}}{\partial\mathbf{X}} - \left(\mathbf{F}_E^T\mathbf{P}\right) : \dot{\mathbf{F}}_P d\mathbf{X} \\ &= \int_{\partial B^0} \mathbf{V}\cdot(\mathbf{P}\mathbf{N})\, ds(\mathbf{X}) - \int_{B^0} \mathbf{V}\cdot\left(\nabla^{\mathbf{X}}\cdot\mathbf{P}\right) + \left(\mathbf{F}_E^T\mathbf{P}\right) : \dot{\mathbf{F}}_P d\mathbf{X}.\end{aligned}$$

Using $R(\mathbf{X},0)\mathbf{A}(\mathbf{X},t) = (\nabla^{\mathbf{X}}\cdot\mathbf{P})(\mathbf{X},t)$ with Equation (2.68) gives

$$\frac{d}{dt}E(t;B^0) = \int_{\partial B^0} \mathbf{V}\cdot(\mathbf{P}\mathbf{N})\, ds(\mathbf{X}) - \int_{B^0} \left(\mathbf{F}_E^T\mathbf{P}\right) : \dot{\mathbf{F}}_P d\mathbf{X} \tag{2.69}$$
$$+ \int_{B^0} \frac{\partial\psi}{\partial\mathbf{F}_P}(\mathbf{F}_E,\mathbf{F}_P) : \dot{\mathbf{F}}_P d\mathbf{X}.$$

Note that $\mathbf{P} = \frac{\partial\psi}{\partial\mathbf{F}_E}(\mathbf{F}_E,\mathbf{F}_P)\mathbf{F}_P^{-T}$, $\left(\mathbf{F}_E^T\mathbf{P}\right) : \dot{\mathbf{F}}_P = \left(\mathbf{F}_E^T\frac{\partial\psi}{\partial\mathbf{F}_E}(\mathbf{F}_E,\mathbf{F}_P)\right) : \left(\dot{\mathbf{F}}_P\mathbf{F}_P^{-1}\right)$. The term

$$\mathbf{L}^P = \dot{\mathbf{F}}_P\mathbf{F}_P^{-1} \tag{2.70}$$

is called the *plastic velocity gradient*. Using this we can write the change in energy as

$$\frac{d}{dt}E(t;B^0) = \int_{\partial B^0} \mathbf{V}\cdot(\mathbf{P}\mathbf{N})\, ds(\mathbf{X}) - \int_{B^0} \mathbf{M}^E : \mathbf{L}^P d\mathbf{X} \tag{2.71}$$
$$+ \int_{B^0} \frac{\partial\psi}{\partial\mathbf{F}_P}(\mathbf{F}_E,\mathbf{F}_P) : \dot{\mathbf{F}}_P d\mathbf{X}.$$

where we define the Mendel stress $\mathbf{M}^E$ as

$$\mathbf{M}^E = \mathbf{F}_E^T\frac{\partial\psi}{\partial\mathbf{F}_E}(\mathbf{F}_E,\mathbf{F}_P). \tag{2.72}$$

The term $\int_{\partial B^0} \mathbf{V}\cdot(\mathbf{P}\mathbf{N})\, ds(\mathbf{X})$ is the rate of work done on B^0 at time t via contact with material external to the region.

2.3.1.2 Isotropy

We assume **the energy density is isotropic**, that is, $\psi(\mathbf{F}_E, \mathbf{F}_P)$ is of the form

$$\psi(\mathbf{F}_E, \mathbf{F}_P) = \hat{\psi}(I(\mathbf{F}_E), II(\mathbf{F}_E), III(\mathbf{F}_E)). \tag{2.73}$$

Then we have

$$\frac{\partial \psi}{\partial \mathbf{F}_E}(\mathbf{F}_E, \mathbf{F}_P) = \alpha \mathbf{F}_E + \beta \mathbf{b}_E \mathbf{F}_E + \gamma \mathbf{F}_E^{-T}$$
$$\boldsymbol{\tau} = \mathbf{P}\mathbf{F}^T = \frac{\partial \psi}{\partial \mathbf{F}_E}(\mathbf{F}_E, \mathbf{F}_P)\mathbf{F}_P^{-T}\mathbf{F}^T = \alpha \mathbf{b}_E + \beta {\mathbf{b}_E}^2 + \gamma \mathbf{I}.$$

Note that $\boldsymbol{\tau}$ and $\mathbf{b}_E$ as well as $\boldsymbol{\tau}$ and ${\mathbf{b}_E}^{-1}$ commute in this case

$$\mathbf{b}_E \boldsymbol{\tau} = \mathbf{b}_E \boldsymbol{\tau}, \;\; {\mathbf{b}_E}^{-1}\boldsymbol{\tau} = \boldsymbol{\tau}{\mathbf{b}_E}^{-1}. \tag{2.74}$$

We can rewrite the plastic dissipation in terms of $\boldsymbol{\tau}$ since

$$\mathbf{M}^E : \mathbf{L}^P = \boldsymbol{\tau} : \left(\mathbf{F}_E \dot{\mathbf{F}}_P \mathbf{F}^{-1}\right). \tag{2.75}$$

Using the definitions in Equations (2.61) and (2.62) and

$$\mathcal{L}_{\mathbf{v}} \mathbf{b}_E \mathbf{b}_E^{-1} = -\mathbf{F}_E \mathbf{F}_P^{-T} \dot{\mathbf{F}}_P^T \mathbf{F}_E^{-1} - \mathbf{F}_E \dot{\mathbf{F}}_P \mathbf{F}^{-1}, \tag{2.76}$$

we can conclude that in the case of isotropic energy density,

$$\begin{aligned}
\boldsymbol{\tau} : \left(\mathcal{L}_{\mathbf{v}} \mathbf{b}_E \mathbf{b}_E^{-1}\right) &= -\boldsymbol{\tau} : \left(\mathbf{F}_E \mathbf{F}_P^{-T} \dot{\mathbf{F}}_P^T \mathbf{F}_E^{-1}\right) - \boldsymbol{\tau} : \left(\mathbf{F}_E \dot{\mathbf{F}}_P \mathbf{F}^{-1}\right) \\
&= -\operatorname{tr}\left(\boldsymbol{\tau} \mathbf{F}_E \mathbf{F}_P^{-T} \dot{\mathbf{F}}_P^T \mathbf{F}_E^{-1}\right) - \boldsymbol{\tau} : \left(\mathbf{F}_E \dot{\mathbf{F}}_P \mathbf{F}^{-1}\right) \\
&= -\operatorname{tr}\left(\boldsymbol{\tau} \mathbf{F}_E \mathbf{F}_E^T \mathbf{F}_E^{-T} \mathbf{F}_P^{-T} \dot{\mathbf{F}}_P^T \mathbf{F}_E^{-1}\right) - \boldsymbol{\tau} : \left(\mathbf{F}_E \dot{\mathbf{F}}_P \mathbf{F}^{-1}\right) \\
&= -\operatorname{tr}\left(\boldsymbol{\tau} \mathbf{b}_E \mathbf{F}^{-T} \dot{\mathbf{F}}_P^T \mathbf{F}_E^{-1}\right) - \boldsymbol{\tau} : \left(\mathbf{F}_E \dot{\mathbf{F}}_P \mathbf{F}^{-1}\right) \\
&= -\operatorname{tr}\left(\mathbf{b}_E \boldsymbol{\tau} \mathbf{F}^{-T} \dot{\mathbf{F}}_P^T \mathbf{F}_E^{-1}\right) - \boldsymbol{\tau} : \left(\mathbf{F}_E \dot{\mathbf{F}}_P \mathbf{F}^{-1}\right) \\
&= -\operatorname{tr}\left(\boldsymbol{\tau} \mathbf{F}^{-T} \dot{\mathbf{F}}_P^T \mathbf{F}_E^{-T}\right) - \boldsymbol{\tau} : \left(\mathbf{F}_E \dot{\mathbf{F}}_P \mathbf{F}^{-1}\right) \\
&= -2\boldsymbol{\tau} : \left(\mathbf{F}_E \dot{\mathbf{F}}_P \mathbf{F}^{-1}\right).
\end{aligned} \tag{2.77}$$

2.3.1.3 Plastic dissipation rate without hardening

In summary, the rate of energy release due to plasticity (with no hardening) can be written as

$$\int_{B^0} \dot{w}_p(\mathbf{X}, t) d\mathbf{X}, \tag{2.78}$$

where

$$\dot{w}_p = \left(\mathbf{F}_E^T \mathbf{P}\right) : \dot{\mathbf{F}}_P = \mathbf{M}^E : \mathbf{L}^p = -\frac{1}{2}\boldsymbol{\tau} : \left(\mathcal{L}_{\mathbf{v}} \mathbf{b}_E \mathbf{b}_E^{-1}\right). \tag{2.79}$$

The last equality only holds for isotropic energy density.

2.3.2 Associative

Assume we have no hardening, e.g. $\tilde{\psi}(\mathbf{F}_E) = \hat{\psi}(\frac{1}{2}\left(\mathbf{F}_E{}^T\mathbf{F}_E - \mathbf{I}\right))$, thus $\mathbf{P} = \mathbf{F}_E \frac{\partial \hat{\psi}}{\partial \mathbf{E}^E}(\frac{1}{2}\left(\mathbf{F}_E{}^T\mathbf{F}_E - \mathbf{I}\right))\mathbf{F}_P^{-T}$ and the Mendel stress $\mathbf{M}^E$ satisfies

$$\mathbf{M}^E = \mathbf{F}_E^T \frac{\partial \psi}{\partial \mathbf{F}_E} = \mathbf{C}_E \frac{\partial \hat{\psi}}{\partial \mathbf{E}^E}. \tag{2.80}$$

If we choose $\mathbf{L}^P$ such that

$$\mathbf{M}^E : \mathbf{L}^P \geq \mathbf{M}^* : \mathbf{L}^P \tag{2.81}$$

for all admissible states of stress $\mathbf{M}^*$, then

1. If $\mathbf{M}^* = \mathbf{0}$ is an admissible state of stress, then

$$\frac{d}{dt} E(t; B^0) \leq \int_{\partial B^0} \mathbf{V} \cdot (\mathbf{P}\mathbf{N})\, ds(\mathbf{X}) \tag{2.82}$$

   which says that the plasticity dissipates energy.

2. If the region of admissible $\mathbf{M}^*$ is (a) convex and (b) defined via $f(\mathbf{M}^*) \leq 0$ then $\mathbf{L}^P \in \partial f(\mathbf{M}^E)$ satisfies Equation (2.81).

Similarly, if we choose $-\frac{1}{2}\mathcal{L}_{\mathbf{v}}\mathbf{b}_E\mathbf{b}_E^{-1} \in \partial f$ we get an associative plastic flow when we write the yield surface in terms of $\boldsymbol{\tau}$: $f(\boldsymbol{\tau})$.

2.3.2.1 Yield surface and plastic flow

We will have plastic flow $\dot{\mathbf{F}}_P \neq \mathbf{0}$ when our stress is on the boundary of the feasible region, and without plasticity we would leave the region. In the case of isoptropy and a yield surface defined in terms of the Kirchhoff stress, then

$$\mathcal{L}_{\mathbf{v}}\mathbf{b}_E = -2\lambda\frac{\partial f}{\partial \boldsymbol{\tau}}(\boldsymbol{\tau})\mathbf{b}_E, \tag{2.83}$$

where

- If $f(\boldsymbol{\tau}) < 0$ or $f(\boldsymbol{\tau}) = 0$ and $\alpha \leq 0$, then $\lambda = 0$.
- Otherwise if, $f(\boldsymbol{\tau}) = 0$ and $\alpha > 0$, then $\lambda = \frac{\alpha}{\beta}$

where

$$\alpha = \frac{\partial f}{\partial \boldsymbol{\tau}} : \frac{\partial \boldsymbol{\tau}}{\partial \mathbf{b}_E} : \dot{\mathbf{b}}_E|_{\dot{\mathbf{F}}_P=\mathbf{0}},\ \beta = 2\frac{\partial f}{\partial \boldsymbol{\tau}} : \frac{\partial \boldsymbol{\tau}}{\partial \mathbf{b}_E} : \left(\frac{\partial f}{\partial \boldsymbol{\tau}}(\boldsymbol{\tau})\mathbf{b}_E\right). \tag{2.84}$$

2.3.2.2 Isoptropic yield surface

Assume **the yield surface function $f : \mathbf{Sym}(3,\mathbb{R}) \to \mathbb{R}$ is isotropic**, that is, $f(\mathbf{V}\boldsymbol{\tau}\mathbf{V}^T) = f(\boldsymbol{\tau})$ for all rotations $\mathbf{V}$. Then as discussed in Section 2.3.4.2, we can write $f(\boldsymbol{\tau}) = \hat{f}(\tau_1, \tau_2, \tau_3)$ where $\boldsymbol{\tau} = \sum_i \tau_i \mathbf{u}_i \otimes \mathbf{u}_i$ and $\frac{\partial f}{\partial \boldsymbol{\tau}}(\boldsymbol{\tau}) = \sum_i \frac{\partial \hat{f}}{\partial \tau_i}\mathbf{u}_i \otimes \mathbf{u}_i$. Therefore since $\boldsymbol{\tau}$ and $\mathbf{b}_E$ have the same eigenvectors

$$\frac{\partial f}{\partial \boldsymbol{\tau}}(\boldsymbol{\tau})\mathbf{b}_E = \sum_i \frac{\partial \hat{f}}{\partial \tau_i}\lambda_{Ei}^2 \mathbf{u}_i \otimes \mathbf{u}_i. \tag{2.85}$$

Furthermore using the properties of isotropic energy density,

$$\beta = 2\sum_{i,j}\frac{\partial \hat{f}}{\partial \tau_i}\tilde{C}_{ij}(\mathbf{b}_E)\frac{\partial \hat{f}}{\partial \tau_j}\lambda_{Ej}^2 \tag{2.86}$$

where

$$\frac{\partial \boldsymbol{\tau}}{\partial \mathbf{b}_E}(\mathbf{b}_E) : \left(\sum_j \sigma_j \mathbf{u}_j \otimes \mathbf{u}_j\right) = \sum_{i,j}\tilde{C}_{ij}(\mathbf{b}_E)\sigma_j \mathbf{u}_i \otimes \mathbf{u}_i$$

for arbitrary $\sum_j \sigma_j \mathbf{u}_j \otimes \mathbf{u}_j$.

2.3.3 Hencky strain

If we define the elastic potential as a function of the Hencky strain as

$$\psi(\mathbf{F}_E, \mathbf{F}_P) = \mu\boldsymbol{\epsilon}_E : \boldsymbol{\epsilon}_E + \frac{\lambda}{2}\mathrm{tr}\left(\boldsymbol{\epsilon}_E\right)^2, \tag{2.87}$$

then

$$\boldsymbol{\tau} = \boldsymbol{C}\boldsymbol{\epsilon}_E = 2\mu\boldsymbol{\epsilon}_E + \lambda\mathrm{tr}\left(\boldsymbol{\epsilon}_E\right)\mathbf{I}. \tag{2.88}$$

This can be written in terms of the eigen basis of $\mathbf{b}_E$ as

$$\boldsymbol{\tau} = \boldsymbol{C}\boldsymbol{\epsilon}_E = \sum_{i,j}\hat{C}_{ij}\log\left(\lambda_j^E\right)\mathbf{u}_i \otimes \mathbf{u}_i \tag{2.89}$$

with

$$[\hat{\mathbf{C}}] = \begin{pmatrix} 2\mu+\lambda & \lambda & \lambda \\ \lambda & 2\mu+\lambda & \lambda \\ \lambda & \lambda & 2\mu+\lambda \end{pmatrix}.$$

2.3.3.1 Yield surface and plastic rate of change

With this energy density, the rate of change of the elastic Hencky strain has the favorable property that its direction is simply related to the yield surface when it is written in terms of $\boldsymbol{\epsilon}_E$. Specifically, α and β in Equation (2.84) can be written as

$$\alpha = \frac{\partial f}{\partial \boldsymbol{\tau}} : \frac{\partial \boldsymbol{\tau}}{\partial \boldsymbol{\epsilon}_E} : \dot{\boldsymbol{\epsilon}}_E|_{\dot{\mathbf{F}}_P=\mathbf{0}},\ \beta = 2\frac{\partial f}{\partial \boldsymbol{\tau}} : \frac{\partial \boldsymbol{\tau}}{\partial \boldsymbol{\epsilon}_E} : \left(\left([\mathbf{B}](\mathbf{b}_E) \circ [\frac{\partial f}{\partial \boldsymbol{\tau}}(\boldsymbol{\tau})\mathbf{b}_E]\right)_{kl}\mathbf{u}_k \otimes \mathbf{u}_l\right), \tag{2.90}$$

and since $\frac{\partial f}{\partial \boldsymbol{\tau}}(\boldsymbol{\tau}) = \sum_i \frac{\partial \hat{f}}{\partial \tau_i}\mathbf{u}_i \otimes \mathbf{u}_i$ and $\mathbf{b}_E = \sum_i \lambda_{Ei}^2\mathbf{u}_i \otimes \mathbf{u}_i$ and

$$[\frac{\partial f}{\partial \boldsymbol{\tau}}(\boldsymbol{\tau})\mathbf{b}_E]_{ij} = \mathbf{u}_i \cdot \left(\frac{\partial f}{\partial \boldsymbol{\tau}}(\boldsymbol{\tau})\mathbf{b}_E\mathbf{u}_j\right) = \begin{cases} \frac{\partial \hat{f}}{\partial \tau_i}\lambda_{Ei}^2, & i = j \\ 0, & \text{otherwise} \end{cases}$$

and $[\mathbf{B}](\mathbf{b}_E)$ from Equation (2.66), as well as $\frac{\partial \boldsymbol{\tau}}{\partial \boldsymbol{\epsilon}_E} = \boldsymbol{C}$ from Equation (2.89)

$$\beta = 2\sum_{i,j}\frac{\partial \hat{f}}{\partial \tau_i}\hat{C}_{ij}\frac{\partial \hat{f}}{\partial \tau_j}. \tag{2.91}$$

2.3.4 Appendix: eigen decomposition differentials

Consider the space of symmetric 3×3 matrices $\mathbb{R}^{3\times3}_{\text{sym}}$, thus $SS \in \mathbb{R}^{3\times3}_{\text{sym}}$ have eigen decompositions, $SS = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$ for some orthogonal $\mathbf{V}$ and diagonal $\mathbf{\Lambda}$. We can define a class of functions $\mathbf{g} : \mathbb{R}^{3\times3}_{\text{sym}} \to \mathbb{R}^{3\times3}_{\text{sym}}$ that are inherited from scalar functions $g : \mathbb{R} \to \mathbb{R}$ as

$$\mathbf{g}(SS) = \mathbf{V}g(\mathbf{\Lambda})\mathbf{V}^T$$

where we use the notation

$$g(\mathbf{\Lambda}) = \begin{pmatrix} g(\lambda_1) & & \\ & g(\lambda_2) & \\ & & g(\lambda_3) \end{pmatrix} \quad \text{and} \quad \mathbf{\Lambda} = \begin{pmatrix} \lambda_1 & & \\ & \lambda_2 & \\ & & \lambda_3 \end{pmatrix}.$$

We can derive the differentials of scalar inherited $\mathbf{g}$ using the expressions for the differentials of the eigen decomposition of SS. The eigen decomposition of the symmetric matrix SS can be thought of as a function over $\mathbb{R}^{3\times3}_{\text{sym}}$: $\mathbf{V} : \mathbb{R}^{3\times3}_{\text{sym}} \to \mathbb{R}^{3\times3}_{\text{orth}}$ and $\mathbf{\Lambda} : \mathbb{R}^{3\times3}_{\text{sym}} \to \mathbb{R}^{3\times3}_{\text{diag}}$, or $\mathbf{V}(SS)$ and $\mathbf{\Lambda}(SS)$ to emphasize the dependent variable. By definition, we have the relation

$$\delta\mathbf{S} = \delta\mathbf{V}\mathbf{\Lambda}\mathbf{V}^T + \mathbf{V}\delta\mathbf{\Lambda}\mathbf{V}^T + \mathbf{V}\mathbf{\Lambda}\delta\mathbf{V}^T$$

and since $\mathbf{V}^T\mathbf{V} = \mathbf{I}$,

$$\delta\mathbf{V}^T\mathbf{V} + \mathbf{V}^T\delta\mathbf{V} = \mathbf{0}.$$

Using $\mathbf{W} = \delta\mathbf{V}^T\mathbf{V}$, we see that $\mathbf{W}$ is skew symmetric and that

$$\mathbf{V}^T\delta\mathbf{S}\mathbf{V} = \mathbf{W}^T\mathbf{\Lambda} + \delta\mathbf{\Lambda} + \mathbf{\Lambda}\mathbf{W}.$$

Since $\mathbf{W}$ is skew symmetric, it can be written as

$$\mathbf{W} = \begin{pmatrix} 0 & \omega_3 & -\omega_2 \\ -\omega_3 & 0 & \omega_1 \\ \omega_2 & -\omega_1 & 0 \end{pmatrix}$$

and thus

$$\mathbf{V}^T\delta\mathbf{S}\mathbf{V} = \begin{pmatrix} \delta\lambda_1 & -\omega_3(\lambda_2-\lambda_1) & \omega_2(\lambda_3-\lambda_1) \\ -\omega_3(\lambda_2-\lambda_1) & \delta\lambda_2 & -\omega_1(\lambda_3-\lambda_1) \\ \omega_2(\lambda_3-\lambda_1) & -\omega_1(\lambda_3-\lambda_2) & \delta\lambda_3 \end{pmatrix}. \tag{2.92}$$

Thus denoting $\mathbf{A} = \mathbf{V}^T\delta\mathbf{S}\mathbf{V}$, we have the expressions

$$\omega_1 = -\frac{a_{32}}{\lambda_3-\lambda_2},\ \omega_2 = \frac{a_{31}}{\lambda_3-\lambda_1},\ \omega_3 = -\frac{a_{21}}{\lambda_2-\lambda_1},\ \text{and } \delta\lambda_i = a_{ii},\ i=1,2,3$$

Similar to the eigen decomposition

$$\mathbf{V}^T\delta\mathbf{g}\mathbf{V} = \mathbf{W}^T g(\mathbf{\Lambda}) + \delta g(\mathbf{\Lambda}) + g(\mathbf{\Lambda})\mathbf{W}$$

where

$$\delta g(\mathbf{\Lambda}) = \begin{pmatrix} g'(\lambda_1)\delta\lambda_1 & & \\ & g'(\lambda_2)\delta\lambda_2 & \\ & & g'(\lambda_3)\delta\lambda_3 \end{pmatrix}.$$

Thus,

$$\mathbf{V}^T\delta\mathbf{g}\mathbf{V} = \begin{pmatrix} g'(\lambda_1)a_{11} & \frac{g(\lambda_2)-g(\lambda_1)}{\lambda_2-\lambda_1}a_{21} & \frac{g(\lambda_3)-g(\lambda_1)}{\lambda_3-\lambda_1}a_{31} \\ \frac{g(\lambda_2)-g(\lambda_1)}{\lambda_2-\lambda_1}a_{21} & g'(\lambda_2)a_{22} & \frac{g(\lambda_3)-g(\lambda_2)}{\lambda_3-\lambda_2}a_{32} \\ \frac{g(\lambda_3)-g(\lambda_1)}{\lambda_3-\lambda_1}a_{31} & \frac{g(\lambda_3)-g(\lambda_2)}{\lambda_3-\lambda_2}a_{32} & g'(\lambda_3)a_{33} \end{pmatrix}$$

and

$$\delta\mathbf{g} = \mathbf{V}\begin{pmatrix} g'(\lambda_1)a_{11} & \frac{g(\lambda_2)-g(\lambda_1)}{\lambda_2-\lambda_1}a_{21} & \frac{g(\lambda_3)-g(\lambda_1)}{\lambda_3-\lambda_1}a_{31} \\ \frac{g(\lambda_2)-g(\lambda_1)}{\lambda_2-\lambda_1}a_{21} & g'(\lambda_2)a_{22} & \frac{g(\lambda_3)-g(\lambda_2)}{\lambda_3-\lambda_2}a_{32} \\ \frac{g(\lambda_3)-g(\lambda_1)}{\lambda_3-\lambda_1}a_{31} & \frac{g(\lambda_3)-g(\lambda_2)}{\lambda_3-\lambda_2}a_{32} & g'(\lambda_3)a_{33} \end{pmatrix}\mathbf{V}^T.$$

We can rewrite this in terms of the matrix

$$\mathbf{B} = \begin{pmatrix} g'(\lambda_1) & \frac{g(\lambda_2)-g(\lambda_1)}{\lambda_2-\lambda_1} & \frac{g(\lambda_3)-g(\lambda_1)}{\lambda_3-\lambda_1} \\ \frac{g(\lambda_2)-g(\lambda_1)}{\lambda_2-\lambda_1} & g'(\lambda_2) & \frac{g(\lambda_3)-g(\lambda_2)}{\lambda_3-\lambda_2} \\ \frac{g(\lambda_3)-g(\lambda_1)}{\lambda_3-\lambda_1} & \frac{g(\lambda_3)-g(\lambda_2)}{\lambda_3-\lambda_2} & g'(\lambda_3) \end{pmatrix}$$

using the Hadamard product (or entry-wise product) where the i,j entry of $\mathbf{A}\circ\mathbf{B}$ is $A_{ij}B_{ij}$ (with no summation on the repeated indices). That is,

$$\delta\mathbf{g} = \mathbf{V}\left(\mathbf{B}\circ\left(\mathbf{V}^T\delta\mathbf{S}\mathbf{V}\right)\right)\mathbf{V}^T \tag{2.93}$$

2.3.4.1 Symmetric tensors

This result generalizes to functions over symmetric tensors. If $\mathbf{g} : \mathcal{V}^2_{\text{sym}} \to \mathcal{V}^2_{\text{sym}}$, then

$$\delta\mathbf{g} = ([\mathbf{B}](SS) \circ [\delta SS])_{kl}\, \mathbf{u}_k \otimes \mathbf{u}_l \tag{2.94}$$

where $SS = \sum_i \lambda_i \mathbf{u}_i \times \mathbf{u}_i$ is the eigenvalue decomposition of SS. $[\delta SS], [\mathbf{B}](SS) \in \mathbb{R}^{3\times3}$ and $[\mathbf{B}](SS) \circ [\delta SS] \in \mathbb{R}^{3\times3}$ is their Hadamard product. The entries in the matrix $[\delta SS]$ are $[\delta SS]_{ij} = \mathbf{u}_i \cdot (\delta SS \mathbf{u}_j)$, i.e. it is the expression of δSS in the eigenbasis of SS. We would assume the convention $\lambda_1 \geq \lambda_2 \geq \lambda_3$ to make the mapping $[\mathbf{B}] :\to \mathcal{V}^2_{\text{sym}}$ well defined from

$$[\mathbf{B}](SS) = \begin{pmatrix} g'(\lambda_1) & \frac{g(\lambda_2)-g(\lambda_1)}{\lambda_2-\lambda_1} & \frac{g(\lambda_3)-g(\lambda_1)}{\lambda_3-\lambda_1} \\ \frac{g(\lambda_2)-g(\lambda_1)}{\lambda_2-\lambda_1} & g'(\lambda_2) & \frac{g(\lambda_3)-g(\lambda_2)}{\lambda_3-\lambda_2} \\ \frac{g(\lambda_3)-g(\lambda_1)}{\lambda_3-\lambda_1} & \frac{g(\lambda_3)-g(\lambda_2)}{\lambda_3-\lambda_2} & g'(\lambda_3) \end{pmatrix}.$$

2.3.4.2 Appendix: Scalar functions of symmetric tensors

Let $f : \mathcal{V}^2_{\text{sym}} \to \mathbb{R}$ with $f(SS) = \hat{f}(\lambda_1, \lambda_2, \lambda_3) = \tilde{f}(I(SS), II(SS), III(SS))$ where

$$I(SS) = \lambda_1 + \lambda_2 + \lambda_3,\ II(SS) = \lambda_1\lambda_2 + \lambda_1\lambda_3 + \lambda_2\lambda_3,\ III(SS) = \lambda_1\lambda_2\lambda_3. \tag{2.95}$$

Using Equation (2.92), we can conclude

$$\delta f = \frac{\partial f}{\partial SS}(SS) = \sum_i \frac{\partial \hat{f}}{\partial \lambda_i}(\lambda_1, \lambda_2, \lambda_3)\delta\lambda_i = \sum_i \frac{\partial \hat{f}}{\partial \lambda_i}(\lambda_1, \lambda_2, \lambda_3)\mathbf{u}_i \cdot (\delta SS \mathbf{u}_i)$$

Thus, the derivative is given by

$$\frac{\partial f}{\partial SS}(SS) = \sum_i \frac{\partial \hat{f}}{\partial \lambda_i}(\lambda_1, \lambda_2, \lambda_3)\mathbf{u}_i \otimes \mathbf{u}_i. \tag{2.96}$$

CHAPTER 3

Simulation and Visualization of Ductile Fracture

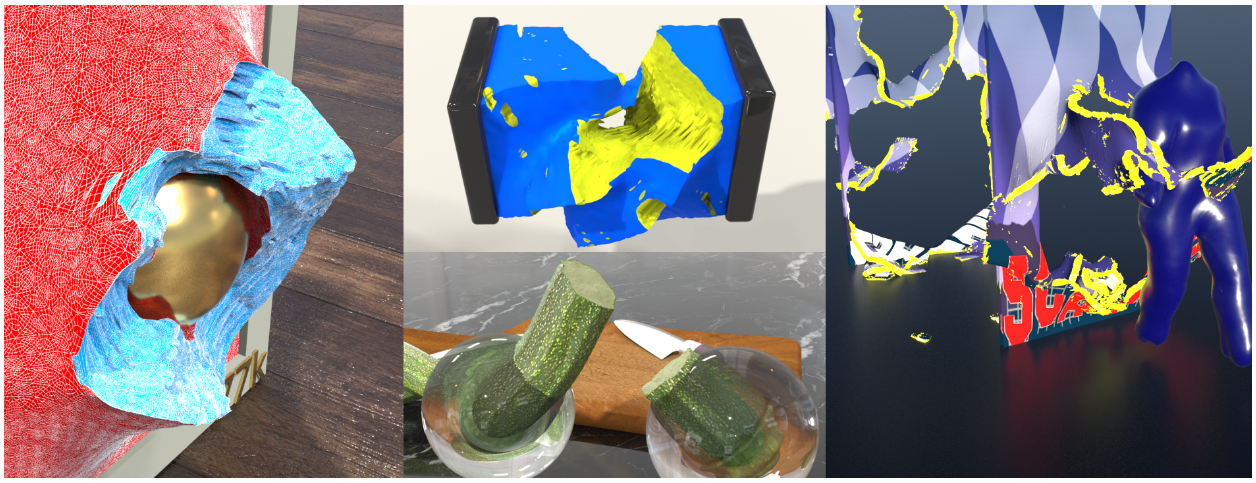

Figure 3.1: **Montage.** Left: Meshing an elastic wall shot by a projectile. Bottom: Breaking a zucchini with brute force. Top: Twisting a cube until it breaks. Right: Ductile walls fracture as a mannequin walks through.

3.1 Introduction

Ductile materials behave elastically until a yield stress condition is met, at which point they yield plastically and at some point fail completely. Whether it be the distinctive patterns exhibited while tearing a piece of fruit or twisted metal after a high-velocity impact, the fracture and failure of ductile materials are ubiquitous and indispensable when creating visually interesting virtual worlds for computer graphics applications. Indeed, some of the earliest methods for simulating elasticity in computer graphics included treatment for tearing and failure of materials [TF88]. O'Brien et al. [OBH02] demonstrated that using the Finite Element Method (FEM) with continual domain remeshing after fracture events allowed for a wide range of ductile behaviors and incredibly detailed simulations. Since this pioneering approach, many others have used FEM and remeshing to achieve similar behaviors [MG04, MBF04, WTG09, WRK10]. Particle methods based on Smoothed Particle Hydrodynamics (SPH) [GBB09, CWX13] and Moving Least Squares (MLS) [PKA05, MKN04] have also been used with impressive effect, since their unstructured nature naturally allows for topological change. Procedural approaches have also achieved good results when computational cost is limited [MHH07, Cho14, JML16].

The Material Point Method (MPM) is another unstructured particle technique that naturally resolves topological changes and fracture, and also naturally accommodates elastoplastic phenomena. Furthermore, a key advantage of MPM is that the hybrid Lagrangian/Eulerian nature of the method naturally resolves collisions between fragments of material. These aspects make MPM an ideal candidate for simulating fracture and failure of ductile materials. However, while MPM naturally allows for topological changes, they can be difficult to control. Particles are connected in the domain when they are in the support of the same Eulerian grid node interpolating function. Particles that do not interact with the same grid nodes in this way are decoupled. This is advantageous in that topology change requires no special treatment; however, fracture is therefore a numerical error that is not influenced by a material property but rather by discretization-related parameters like particle sampling density and Eulerian grid resolution.

Numerical fracture can be addressed by utilizing particle resampling techniques as in [YSB15] or by using the Lagrangian energy technique of Jiang et al. [JSS15] in which a tetrahedron mesh is used to compute deformation gradients. This treatment naturally couples meshed objects with MPM-based materials, and also gives an automated treatment of self-collision between meshed objects and other materials. However, in either the resampling

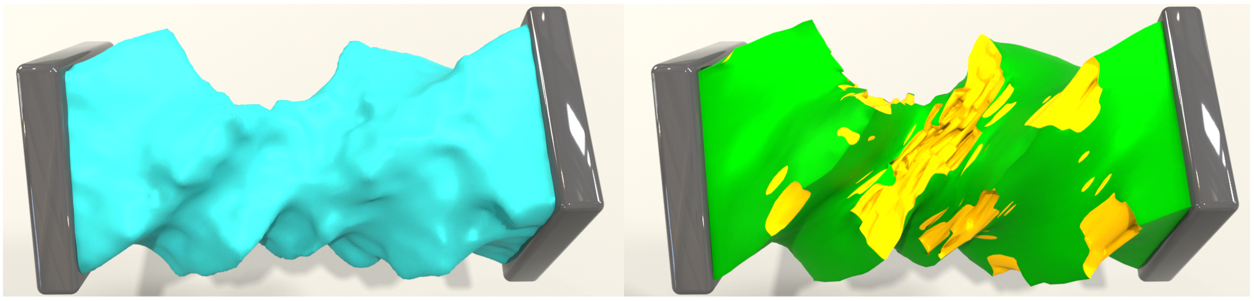

Figure 3.2: **Comparison of mesh visualization using twisted cube example.** A cube with 8,000 particles was twisted to fracture in the simulation. We render the results with Houdini particle fluid surface (left) and our mesh visualization (right).

or Lagrangian energy approaches, an additional model must be provided to allow for fracture.

A second issue hindering MPM adoption for ductile fracture is largely common to all particle-based techniques: defining and rendering material boundary surfaces in a visually sharp manner is difficult. While particle-based simulation techniques naturally allow for topological change, they generally have a more vague notion of material boundaries that complicates the process of rendering. FEM and mesh-based techniques require more intervention (remeshing) to resolve topological change, however in the process material boundaries are sharp and well defined. This is important for preserving the surface of objects created by users, and for transferring textures as the material fails.

The most common techniques for visualizing particle-based simulation data define the boundary of the particle domain as the zero isocontour of a level set function, or as a threshold value of a density function. This goes back to at least Blinn [Bli82]. Many other authors have provided improvements on these techniques over the years, including sharper surface resolution, reduction of noise and temporal coherence of surfaces, resolution of anisotropic features, and many more [MCG03, ZB05, SSP07, APK07, YT13, ATW13, MCZ07, Mus14]. However, these types of techniques are much more appropriate for fluid simulations, and

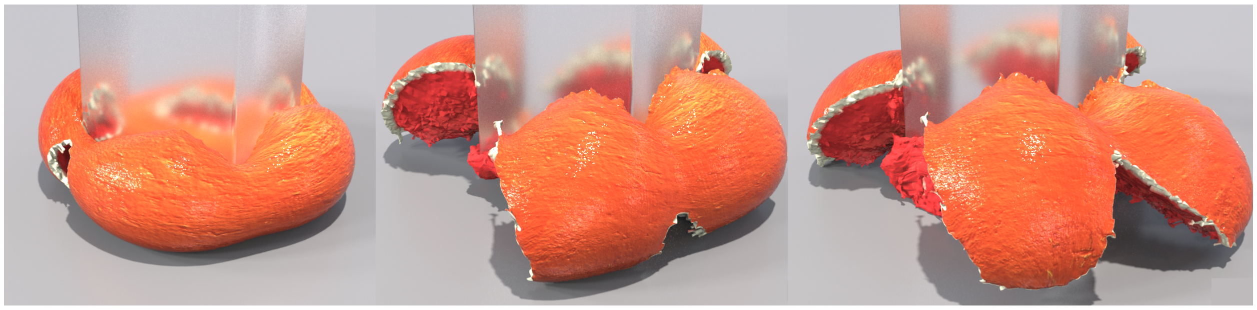

Figure 3.3: **Hydraulic press.** The orange is simulated with a meshed hollow sphere filled with guts made by MPM particles.

cannot support initialization from a high-resolution textured input surface mesh without complicated texture transfer at each frame, etc.

Surface tracking techniques can provide the desired preservation of sharp features and surface details. These techniques have been used with great effect in simulations of fluid [BB09, DBG14, M09, WTG10, YWT12] and viscoelastic materials [WTG09, DGP17]. These approaches are extremely powerful, but computationally expensive. However, much of the implementation and computational overhead is associated with material merging. Much simpler techniques can be used if only splitting is required. Fracture of ductile materials typically only involves failure without cohesive merging, so fully-general surface tracking techniques are not necessary.

Pre-scoring-based surfacing approaches are generally more efficient than surface tracking, and can be used when merging is not needed. These techniques predefine the maximally split configuration of the material, and only separation between components can occur. For example, the virtual node algorithm of Molino et al. [MBF04] is a pre-scoring technique where each vertex in a tetrahedron mesh represents a portion of the material in the elements in its one ring. Choi [Cho14] use a pre-scoring approach for visualizing shape-matching-based ductile fracture where each node is assigned material as a union of elements, gathered via

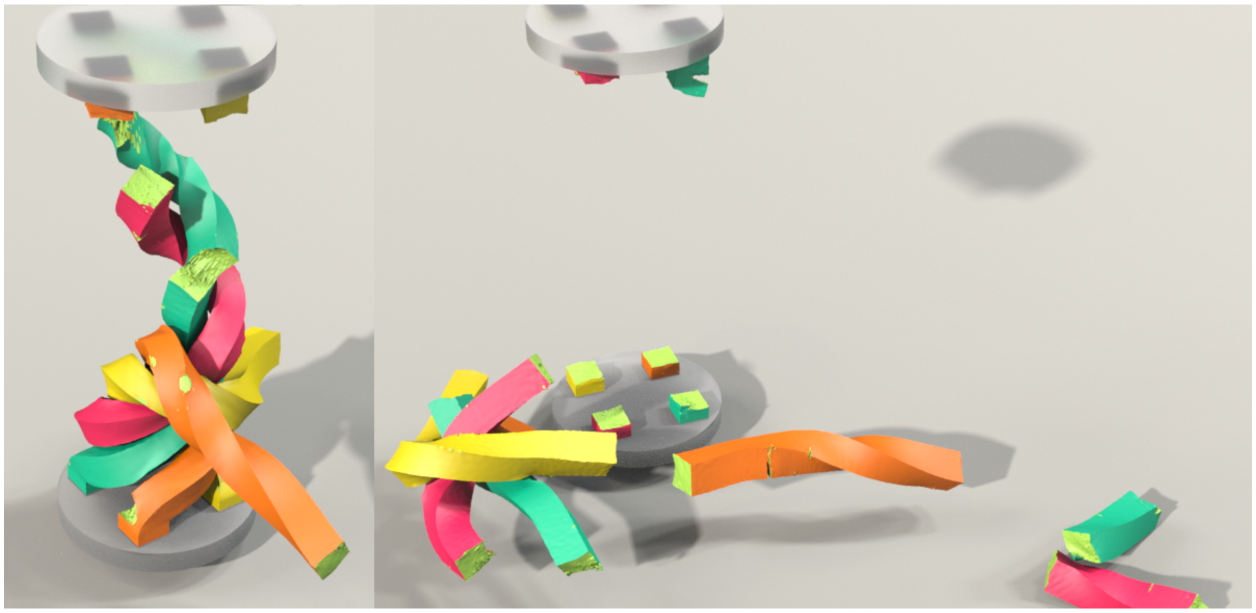

Figure 3.4: **Four columns braided to fracture.**

K-means, from a tetrahedron mesh. Chen et al. [CZZ18] assign a single tetrahedron to each particle by initializing particles at the barycenters of an input tetrahedron mesh. In these techniques, material separation is introduced when connectivity between adjacent particle regions is severed. Crack surfaces are then defined as a subset of the boundary of the maximally split configuration. Generally, pre-scoring techniques suffer from mesh-based aliasing, since the crack paths must lie on the predefined maximally split configuration. Fracture surfaces are usually much smoother than they will appear when the sampling bias in the predefined maximally split configuration is imposed on the visualization.

We provide two options to remove the barriers preventing MPM adoption for ductile fracture simulation in graphics applications. First, we provide an extension of the mesh based strategy of Jiang et al. [JSS15] that removes numerical fracture and introduces failure through the elastoplastic constitutive equations alone. Second, when traditional particle-based MPM with numerical fracture suffices, we overcome limitations of existing surfacing strategies with a pre-scoring approach. We note that our surfacing approach is a post-process that can be

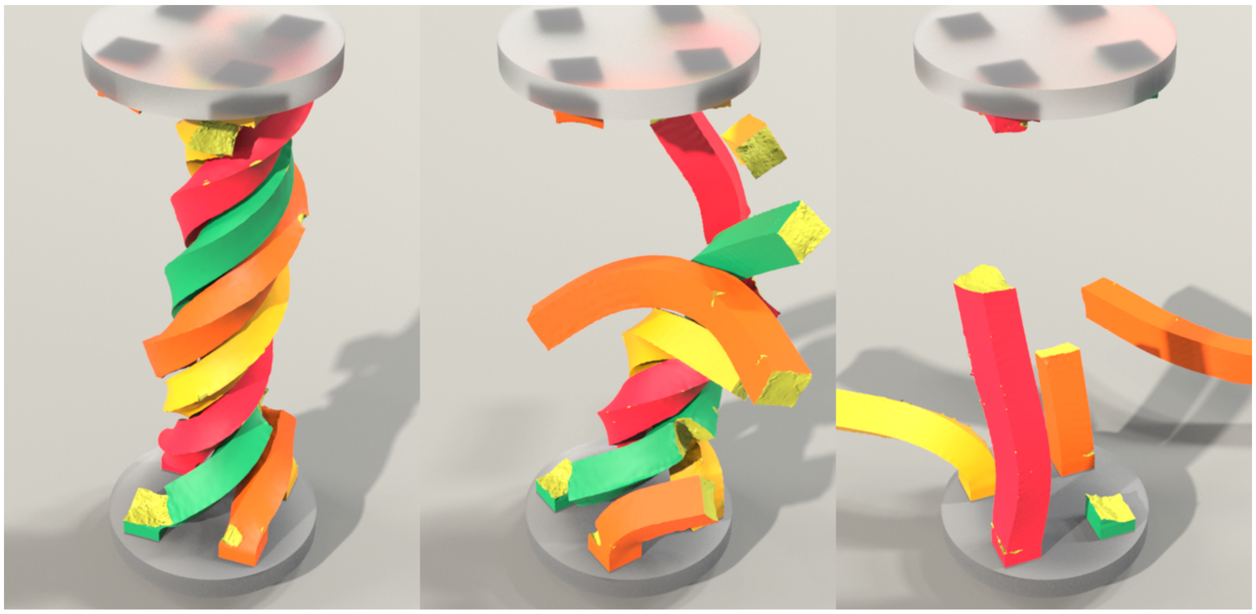

Figure 3.5: **Four stiffer columns braided to fracture.**

implemented on data generated from standard MPM simulations. In summary, our contributions include:

- An elastoplasticity and damage model for ductile fracture that works easily with existing MPM code bases.
- A generalization of the Lagrangian energy approach of Jiang et al. [JSS15] for removing numerical fracture with ductile materials.
- A novel particle surfacing technique that preserves input surface details like texture and high-curvature regions, while removing mesh-based aliasing inherent in pre-scoring surfacing strategies.

3.2 Previous work

Here we discuss works from the computer graphics and computational physics literature related to simulation of ductile fracture and visualization of particle-based simulation data.

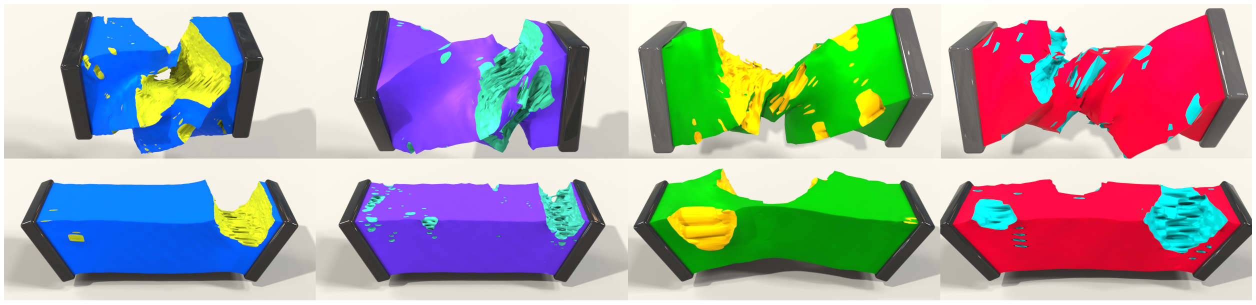

Figure 3.6: **Twisting and Pulling.** Four identical cubes of different resolution undergoing twisting and pulling motions. From left to right: 60K, 17K, 8K, 4K particles.

Following the seminal approach of O'Brien et al. [OBH02], many authors have used FEM simulation of elastoplasticity with continual domain remeshing for ductile fracture. Müller et al. [MG04] use warped stiffness with a Rankine condition on the principal stress to define per-tetrahedron element fracture planes. Pfaff et al. [PNJ14] use an adaptive mesh to simulate tearing and cracking of thin sheets. Parker and O'Brien [PO09] use the separation tensor from [OH99] but split along element boundaries rather than cutting elements for the sake of efficiency. Wicke et al. [WRK10] dynamically remesh tetrahedron meshes to allow for efficient simulation of behaviors ranging from purely elastic to extremely plastic with fracture. Other remeshing approaches include [BWH07, WT08, WTG09, BDW13]. Wicke et al. [WBG07, KMB08] developed interpolating functions for convex polyhedral elements to allow for easy splitting of elements in fracture simulations. Gissler et al. [GBT07] introduce a notion of constraint sets for fracture simulation. Koschier et al. [KBT17] use XFEM and improve the mass matrix treatment by integrating over partially empty enriched elements. Zhang et al. [ZZS06] use tetrahedron mesh-based FEM with elastoplasticity driven damage, element splitting (at damage threshold), and molecular dynamics for debris simulation.

Pauly et al. [PKA05] use a meshfree MLS approach to simulate elastoplastic ductile fracture with Heaviside-enriched interpolating functions, as in the XFEM approaches of Belytschko

[BCX03]. They create domain and crack boundary surfaces at render time using the surfels approach in [PKK03, WTG04]. Müller et al. [MKN04] use a similar approach. Steineman et al. [SOG09] use visibility graphs to further improve the modification of MLS interpolating functions in the presence of splitting and merging defined by explicitly tracked failure surfaces. Gerszewski et al. [GBB09] also compute the deformation gradient in a weighted least squares sense.

Other notable ductile fracture techniques include the peridynamics approach of Chen et al. [CZZ18]. Bußler et al. [BDP17] visualize crack surfaces in peridynamics particle data by computing Delaunay tetrahedralizations that respect height ridges in the damage field. Choi [Cho14] uses shape-matching to simulate procedural ductile fracture. Ohta et al. [OKN09] use an adaptive regular lattice with shape matching-based elasticity to simulate ductile fracture. Jones et al. [JML16] simulate ductile fracture using shape matching.

Various approaches for ductile fracture with MPM exist in the computational physics literature. Wretborn et al. [WAM17] simulate fracture with MPM by pre-scoring materials into pieces held together by massless particle constraints. They resolve collisions between fragments by using the MPM N-body approach of [HZM11]. Nairn et al. [Nai03, GN06] developed the CRAMP MPM technique for simulating velocity and displacement discontinuities on the grid. Other MPM techniques utilize grid node duplication [DLC07]. They then resolve frictional contact on the duplicated Eulerian grid nodes.

Surfacing particle-based simulation data is a long-standing problem. Most approaches define the boundary of the particle domain as the zero isocontour of a level set function or as a threshold value of a density function [Bli82, DC98, MCG03, ZB05, APK07, SSP07, Mus14]. Yu and Turk developed an anisotropic approach to more accurately capture sharp features [YT13]. Bhattacharya et al. [BGB15] fit signed distance functions to particle data by minimizing a biharmonic thin shell energy over a surface constrained between interior and exterior CSG surfaces, and support anisotropic capture of sharp features as in [YT13]. Williams [Wil08] similarly solves the surfacing problem with a constrained minimization. Shen and Shah [SS07] address temporal discontinuities by blending adjacent frames. Museth et al. [MCZ07] incorporate a variety of post-processing techniques including temporal and spatial anti-aliasing. Adams et al. [APK07] use a semi-Lagrangian contouring method similar to that proposed by Bargteil et al. [BGO06]. Dagenais et al. [DGP17] improves and extends surface tracking to retain surface details. Mercier et al. [MBT15] develop a post-process approach for surfacing particle-based fluid simulation data. They create an up-res particle surface using a generalization of the approach in [Wil08] and then apply a surface-only Lagrangian wave simulation to provide realistic, detailed motion.

Pre-scoring bodies into precomputed pieces is useful for simulation and visualization. Müller et al. [MCK13] decompose objects into convex pieces and generate fracture patterns of space using Voronoi diagrams. CSG operations are used to resolve the initial convex decomposition with the fracture patterns. Su et al. [SSF09] also fracture all of space to generate rigid body fragment pieces for real time simulation of brittle fracture. Liu et al. [LHL11] also pre-score the material along Voronoi boundaries to add user control over fracture patterns. Schvartzman and Otaduy [SO14] use Voronoi-based pre-scoring of fracture boundaries with rigid body simulation to simulate brittle fracture. Zheng and James [ZJ10] use the strain energy density to adapt Voronoi fracture regions. Raghavachary [Rag02] defines fragments in polygon meshes by splitting into Voronoi regions.

3.3 Mathematical models

We define the deformation of a continuum body as a map from its undeformed configuration consisting of points $\mathbf{X}$ to its deformed configuration consisting of points $\mathbf{x}$ at time t by $\mathbf{x}(t) =$

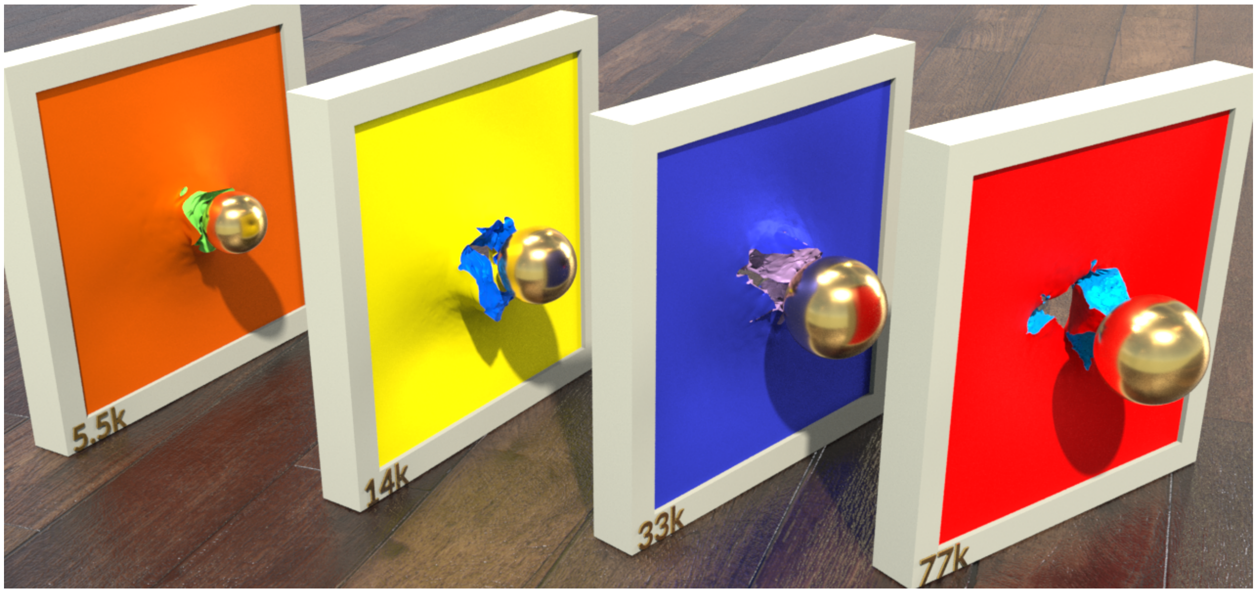

Figure 3.7: **Projectiles and thin walls.** Shooting projectiles at ductile walls with 5.5K (orange), 14K (yellow), 33K (blue), and 77K (red) particles.

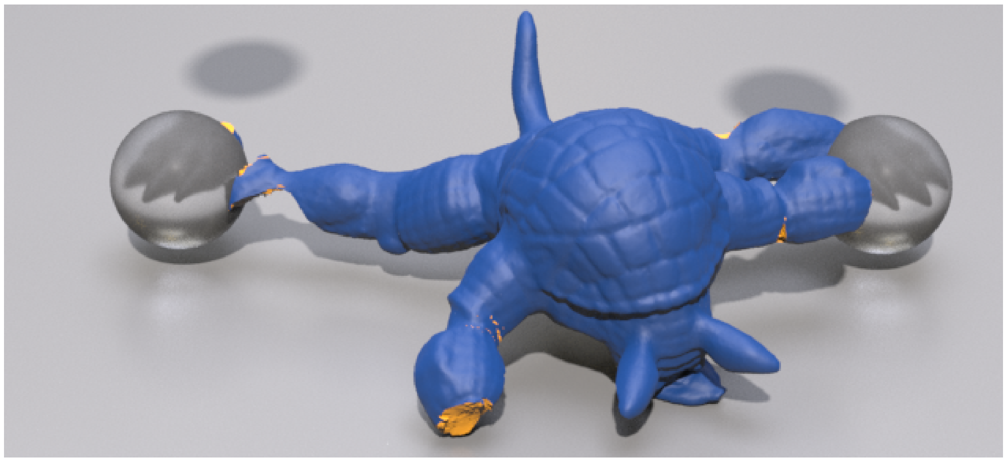

Figure 3.8: **Stretching armadillo.** An armadillo stretched to fracture.

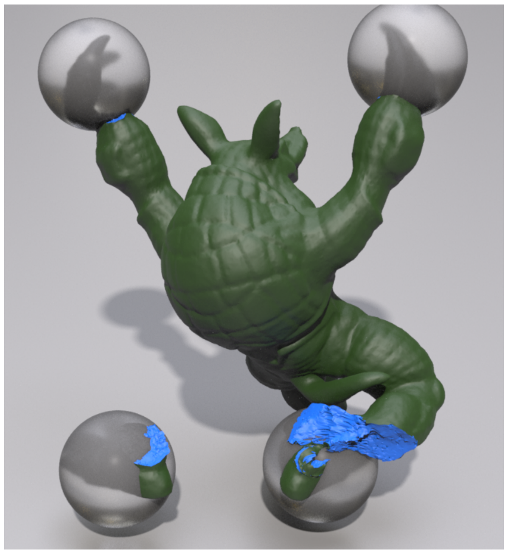

Figure 3.9: **Twisting armadillo.** An armadillo twisted to fracture.

$\phi(\mathbf{X}, t)$. We refer to the spatial derivative of this map as the deformation gradient $\mathbf{F} = \frac{\partial \phi}{\partial \mathbf{X}}$ and decompose it into elastic and plastic parts $\mathbf{F} = \mathbf{F}^E \mathbf{F}^P$. Here $\mathbf{F}^E$ is the elastic deformation and $\mathbf{F}^P$ is the plastic deformation associated with inelastic yielding at large stresses [BW08]. The potential energy in the system increases as $\mathbf{F}^E$ deviates from orthogonality, meaning that the motion from the plastic/damaged state is non-rigid. The governing equations for the deformation mapping are derived from conservation of mass and momentum

$$\frac{D\rho}{Dt} = -\rho \nabla \cdot \mathbf{v}, \tag{3.1}$$

$$\rho \frac{D\mathbf{v}}{Dt} = \nabla \cdot \boldsymbol{\sigma} + \rho \mathbf{g}, \tag{3.2}$$

where $\boldsymbol{\sigma}$ denotes the Cauchy stress, $\mathbf{g}$ the gravity, and $\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{v} \cdot \nabla$ the material derivative.

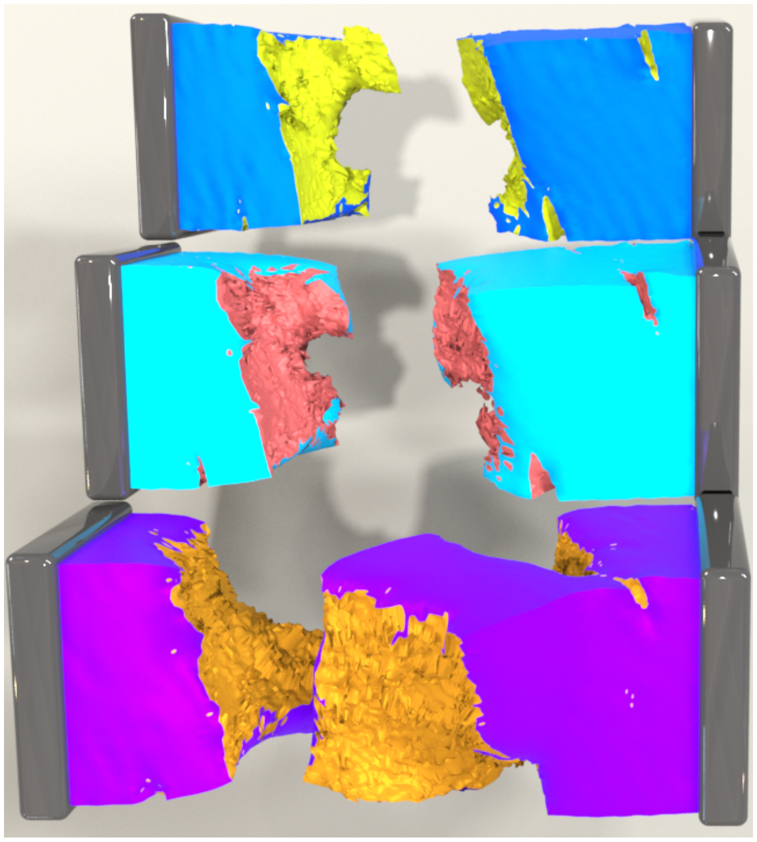

Figure 3.10: **Twisting with von Mises.** Twisting cubes with different von Mises yield surfaces. We use $\tau_C = E$, (blue), $0.7E$ (cyan), and $0.5E$, for Young's modulus E.

3.3.1 Elastic constitutive model

We use the isotropic hyperelastic potential energy density of [KGP16]. This model is quadratic in elastic Hencky strain $\boldsymbol{\epsilon}^E = \frac{1}{2}\ln(\mathbf{F}_E\mathbf{F}_E^T)$,

$$\psi(\mathbf{F}^E) = \mu\boldsymbol{\epsilon} : \boldsymbol{\epsilon} + \frac{\lambda}{2}\text{tr}(\boldsymbol{\epsilon})^2 = \mu\sum_{i=1}^{3}\ln(\sigma_i^E)^2 + \frac{\lambda}{2}\left(\sum_{i=1}^{3}\ln(\sigma_i^E)\right)^2 \tag{3.3}$$

where $\mathbf{F}^E = \mathbf{U}^E\boldsymbol{\Sigma}^E(\mathbf{V}^E)^T$ is the singular value decomposition of $\mathbf{F}_E$ and σ_i^E denote the entries in $\boldsymbol{\Sigma}^E$. Here μ and λ are the Lamé coefficients which control the amount of resistance to deformation and volume change. The Cauchy stress is defined in terms of the elastic potential as

$$\boldsymbol{\sigma} = \frac{1}{\det(\mathbf{F})}\frac{\partial\psi}{\partial\mathbf{F}_E}{\mathbf{F}_E}^T, \tag{3.4}$$

$$\frac{\partial\psi}{\partial\mathbf{F}^E} = \mathbf{U}^E{\boldsymbol{\Sigma}^E}^{-1}\left(2\mu\ln(\boldsymbol{\Sigma}^E) + \lambda\ln(\boldsymbol{\Sigma})\right)(\mathbf{V}^E)^T. \tag{3.5}$$

This choice of potential energy is primarily for the sake of simplifying the return mapping process (see [WDG19a]), as discussed in [KGP16, JGT17].

3.3.2 Plasticity

Ductile materials behave elastically until a critical stress is reached, at which point deformation becomes permanent and the material achieves a new local rest state. We express this notion of critical stress in terms of a yield surface in stress space defined implicitly as $y(\boldsymbol{\sigma}) = 0$ using a yield function y. When $y(\boldsymbol{\sigma}) < 0$, the critical stress has not been achieved and the material behaves elastically. When $y(\boldsymbol{\sigma}) = 0$, the elastic limit is reached and the plastic deformation defined via $\mathbf{F}^P$ becomes non-trivial. Mathematically, we can view the dynamics of $\mathbf{F}^P$ as being chosen to satisfy the stress constraint $y(\boldsymbol{\sigma}) = 0$ through its dependence on $\mathbf{F}^E$.

Although the Cauchy stress $\boldsymbol{\sigma}$ is more physically intuitive, the Kirchhoff stress $\boldsymbol{\tau} = \det(\mathbf{F})\boldsymbol{\sigma}$ is often more convenient when working with plasticity. It is particularly convenient for defin-

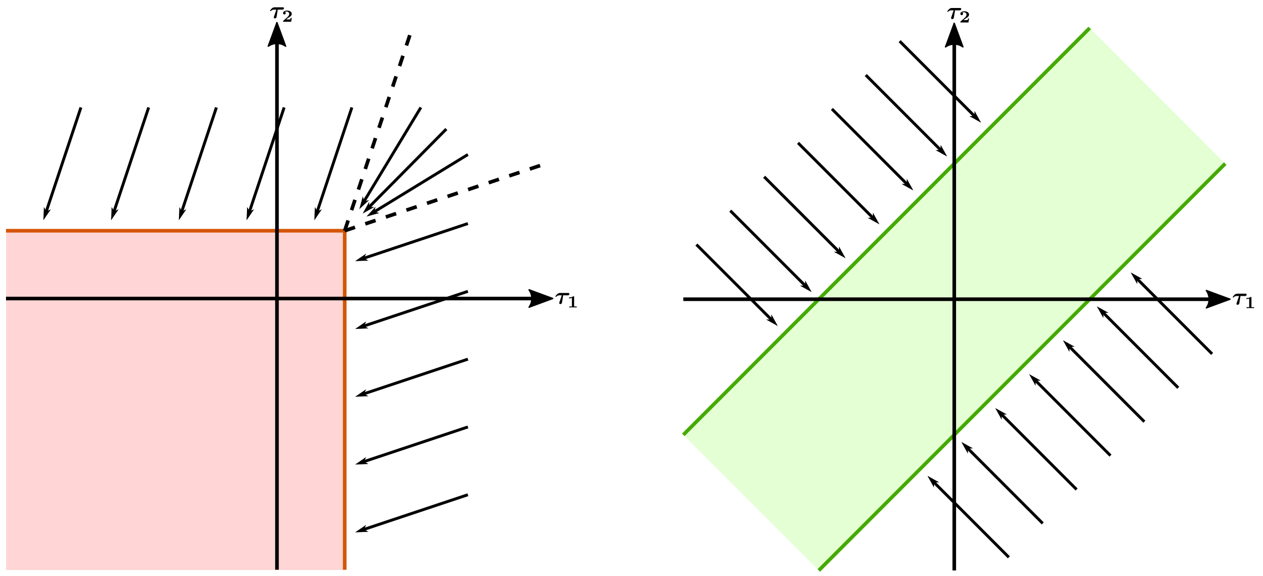

Figure 3.11: **Return mappins.** Left: Rankine yield surface and its return mapping. Right: von Mises yield surface and its return mapping.

ing the plastic deformation in a manner that is consistent with the second law of thermodynamics and when enforcing the yield condition discretely during time stepping, a process which is typically referred to as the return mapping (see [WDG19a]). Henceforth, we will assume the yield surface is defined in terms of the Kirchhoff stress $y(\boldsymbol{\tau})$.

3.3.2.1 Yield surface

We use two different yield surfaces to model different fracture modes. The Rankine yield surface [And17] is given by

$$y(\boldsymbol{\tau}) = \max_{\|\mathbf{u}\|=\|\mathbf{v}\|=1} \mathbf{u}^T \boldsymbol{\tau} \mathbf{v} - \tau_C \leq 0, \tag{3.6}$$

where τ_C is a scalar parameter that represents the maximum allowed tensile strength, since the expression $\max_{\|\mathbf{u}\|=\|\mathbf{v}\|=1} \mathbf{u}^T \boldsymbol{\tau} \mathbf{v}$ measures the tensile stress among all directions and corresponds to the largest eigenvalue of $\boldsymbol{\tau}$. Constraining the maximal tension in all directions enables the material to go through mode I yielding, where permanent deformation is induced in response to local tension.

The von Mises yield surface given by

$$y(\boldsymbol{\tau}) = \|\boldsymbol{\tau} - \operatorname{tr}(\boldsymbol{\tau})\mathbf{I}\|_F - \tau_C \leq 0 \tag{3.7}$$

provides plastic response to mode II and mode III shearing deformations by constraining the deviatoric (shear) stress; here $\|\mathbf{A}\|_F = \sqrt{\mathbf{A} : \mathbf{A}}$ denotes the Frobenius norm. By combining the two yield surfaces or using them independently, we can simulate a wide range of fracturing and plastic materials.

In practice, the yield condition $y(\boldsymbol{\tau}) \leq 0$ is enforced per time step. In this process, the trial strain ($\tilde{\boldsymbol{\epsilon}}^E$) is mapped from a state whose corresponding stress violates the condition to one whose corresponding stress is on the boundary of the yield surface ($\boldsymbol{\epsilon}^{E,n+1}$) in a process referred to as the return mapping. We illustrate the different yield surfaces and the associative direction for return mappings in Figure 3.11. We provide detailed derivation in Section 3.3.2.2

3.3.2.2 Return mapping

A trial state of deformation $\tilde{\mathbf{F}}^E$ is computed, assuming no plastic flow from time t^n to t^{n+1}. With this assumption, the plastic deformation does not change over the time step, so $\mathbf{F}^{P,n+1} = \mathbf{F}^{P,n}$, and $\mathbf{F}^{E,n+1} = \tilde{\mathbf{F}}^E$. However, if the yield condition is violated when $\boldsymbol{\tau}$ is computed from the trial deformation $\tilde{\mathbf{F}}^E$, then $\tilde{\mathbf{F}}^E$ must be modified accordingly to satisfy the constraint. This process is often referred to as the return mapping: $\tilde{\mathbf{F}}^E \to \mathbf{F}^{E,n+1}$. There are infinitely many ways that this can be done. We use associative plastic flow since it is straightforward with our choice of hyperelastic potential, and guarantees no violation of the second law of thermodynamics. See Section 2.3.2 for details.

Associativity requires that the projection of the stress be done in a direction equal to the elasticity tensor $\mathcal{C} = 2\mu\mathcal{I} + \lambda\mathbf{I}\otimes\mathbf{I}$ times the normal to the yield surface $\frac{\partial y}{\partial\boldsymbol{\tau}}$. Here $\mathcal{C}$ is a fourth-order tensor, $\mathcal{I}$ the fourth-order identity tensor, and $\mathbf{I}$ the second-order identity tensor. This process can be described succinctly in terms of the trial and project elastic Hencky strain as

$$\tilde{\boldsymbol{\epsilon}}^E - \boldsymbol{\epsilon}^{E,n+1} = \delta\frac{\partial y}{\partial\boldsymbol{\tau}}(\mathcal{C} : \boldsymbol{\epsilon}^{E,n+1}), \tag{3.8}$$

where $\tilde{\boldsymbol{\epsilon}}^E = \frac{1}{2}\ln(\tilde{\mathbf{F}}^E(\tilde{\mathbf{F}}^E)^T)$ is the trial elastic Hencky strain, $\boldsymbol{\epsilon}^{E,n+1} = \frac{1}{2}\ln(\mathbf{F}^{E,n+1}(\mathbf{F}^{E,n+1})^T)$ is the projected elastic Hencky strain, $\mathcal{C} : \boldsymbol{\epsilon}^{E,n+1} = \boldsymbol{\tau} = \lambda\mathrm{tr}(\boldsymbol{\epsilon}^{E,n+1})\mathbf{I} + 2\mu\boldsymbol{\epsilon}^{E,n+1}$ is the elasticity tensor, and $\delta > 0$ is a Lagrange multiplier chosen so that $\boldsymbol{\epsilon}^{E,n+1}$ is on the yield surface.

Due to our assumption of isotropy, the constraint in Equation (3.8) can be satisfied in terms of the singular values of the elastic deformation gradient. Furthermore, the singular vectors of the trial elastic strain do not change in the return mapping:

$$\mathbf{F}^{E,n+1} = \mathbf{U}^E\boldsymbol{\Sigma}^{E,n+1}(\mathbf{V}^E)^T,\ \tilde{\mathbf{F}}^E = \mathbf{U}^E\tilde{\boldsymbol{\Sigma}}^E(\mathbf{V}^E)^T. \tag{3.9}$$

With this convention, the trial and projected Hencky strains and Kirchhoff stresses satisfy

$$\tilde{\boldsymbol{\epsilon}}^E = \mathbf{U}^E\ln\tilde{\boldsymbol{\Sigma}}^E(\mathbf{U}^E)^T \tag{3.10}$$

$$\boldsymbol{\epsilon}^{E,n+1} = \mathbf{U}^E\ln\boldsymbol{\Sigma}^{E,n+1}(\mathbf{U}^E)^T \tag{3.11}$$

and

$$\tilde{\boldsymbol{\tau}}^E = \mathbf{U}^E\left(\lambda\mathrm{tr}(\ln(\tilde{\boldsymbol{\Sigma}}^E))\mathbf{I} + 2\mu\ln(\tilde{\boldsymbol{\Sigma}}^E)\right)(\mathbf{U}^E)^T \tag{3.12}$$

$$\boldsymbol{\tau}^{E,n+1} = \mathbf{U}^E\left(\lambda\mathrm{tr}(\ln(\boldsymbol{\Sigma}^{E,n+1}))\mathbf{I} + 2\mu\ln(\boldsymbol{\Sigma}^{E,n+1})\right)(\mathbf{U}^E)^T, \tag{3.13}$$

respectively.

The return mapping is completed as an operation on the eigenvalues $\tilde{\boldsymbol{\epsilon}}^E$. For simplicity of notation, we henceforth denote the eigenvalues of $\tilde{\boldsymbol{\epsilon}}^E$ and $\tilde{\boldsymbol{\tau}}^E$ by $\hat{\boldsymbol{\epsilon}}$ and $\hat{\boldsymbol{\tau}} = \lambda(\mathbf{1}\cdot\hat{\boldsymbol{\epsilon}})\mathbf{1} + 2\mu\hat{\boldsymbol{\epsilon}}$ respectively, where $\mathbf{1}$ is the vector of all ones. Furthermore, we refer to the eigenvalues of the projected $\boldsymbol{\epsilon}^{E,n+1}$ and $\boldsymbol{\tau}^{E,n+1}$ as $\mathrm{proj}(\hat{\boldsymbol{\epsilon}})$ and $\mathrm{proj}(\hat{\boldsymbol{\tau}})$ respectively. The process of satisfying Equation (3.8) is, in the case of the Rankine yield condition,

- If $\lambda\mathbf{1}\cdot\hat{\boldsymbol{\epsilon}} + 2\mu\epsilon_1 \leq \tau_C$, no projection, $\mathrm{proj}(\hat{\boldsymbol{\epsilon}}) = \hat{\boldsymbol{\epsilon}}$,
- If $(2\mu+\lambda)\epsilon_2 + \lambda(\mathbf{1}\cdot\hat{\boldsymbol{\epsilon}} - \epsilon_1) \leq \tau_C < \lambda\mathbf{1}\cdot\hat{\boldsymbol{\epsilon}} + 2\mu\epsilon_1$, $\mathrm{proj}(\hat{\boldsymbol{\epsilon}}) = \left(\frac{\tau_C - \lambda(\mathbf{1}\cdot\hat{\boldsymbol{\epsilon}}-\epsilon_1)}{2\mu+\lambda}, \epsilon_2, \epsilon_3\right)$,
- If $(2\mu+3\lambda)\epsilon_3 \leq \tau_C < (2\mu+\lambda)\epsilon_2+\lambda(\mathbf{1}\cdot\hat{\boldsymbol{\epsilon}}-\epsilon_1)$, $\mathrm{proj}(\hat{\boldsymbol{\epsilon}}) = \left(\frac{\tau_C-\lambda(\mathbf{1}\cdot\hat{\boldsymbol{\epsilon}}-\epsilon_1-\epsilon_2)}{2\mu+2\lambda}, \frac{\tau_C-\lambda(\mathbf{1}\cdot\hat{\boldsymbol{\epsilon}}-\epsilon_1-\epsilon_2)}{2\mu+2\lambda}, \epsilon_3\right)$,
- If $\tau_C < (2\mu+3\lambda)\epsilon_3$, $\mathrm{proj}(\hat{\boldsymbol{\epsilon}}) = \frac{\tau_C}{2\mu+3\lambda}\mathbf{1}$.

In the case of the von Mises yield condition, the projection is

- If $|\hat{\boldsymbol{\tau}} - \mathbf{1}\cdot\hat{\boldsymbol{\tau}}\mathbf{1}| \leq \tau_C$, no projection, $\mathrm{proj}(\hat{\boldsymbol{\epsilon}}) = \hat{\boldsymbol{\epsilon}}$,
- If $|\hat{\boldsymbol{\tau}} - \mathbf{1}\cdot\hat{\boldsymbol{\tau}}\mathbf{1}| > \tau_C$, $\mathbf{p} = (\hat{\boldsymbol{\tau}}\cdot\mathbf{1})\frac{\mathbf{1}}{3}$, $\mathbf{d} = \hat{\boldsymbol{\tau}} - \mathbf{p}$, $\mathrm{proj}(\hat{\boldsymbol{\tau}}) = \mathbf{p} + \tau_C\frac{\mathbf{d}}{|\mathbf{d}|}$, $\mathrm{proj}(\hat{\boldsymbol{\epsilon}}) = \hat{\mathcal{C}}^{-1}\mathrm{proj}(\hat{\boldsymbol{\tau}})$

where

$$\hat{\mathcal{C}} = \begin{pmatrix} 2\mu+\lambda & \lambda & \lambda \\ \lambda & 2\mu+\lambda & \lambda \\ \lambda & \lambda & 2\mu+\lambda \end{pmatrix}. \tag{3.14}$$

After the projection has been done, the singular values of the time t^{n+1} elastic deformation gradient are computed from $\boldsymbol{\Sigma}^{E,n+1} = \exp(\mathrm{proj}(\hat{\boldsymbol{\epsilon}}))$, which are used to construct the deformation gradient as in Equation (3.9). Lastly, the time t^{n+1} plastic deformation gradient is computed from $\mathbf{F}^{P,n+1} = (\mathbf{F}^{E,n+1})^{-1}\mathbf{F}^{n+1}$.

3.3.2.3 Softening and damage

As the material undergoes plastic deformation, we decrease τ_C to shrink the yield surface towards the origin. This limits the strength of the material as smaller and smaller stresses are admissible. For each projection $\tilde{\boldsymbol{\epsilon}}^E \rightarrow \boldsymbol{\epsilon}^{E,n+1}$ in the return mapping (see [WDG19a]), we decrease τ_C by $\theta\|\boldsymbol{\epsilon} - \mathrm{proj}(\boldsymbol{\epsilon})\|_F$, where $\theta > 0$ is a material constant that defines the rate of softening. When τ_C reaches zero, we model the material as completely damaged and set the Lamé coefficients to zero.

3.4 Numerical method

We use MPM to discretize the governing equations and cover both standard particle-based MPM as in [SCS94, SSC13] as well as the mesh-based Lagrangian energy techniques used to prevent numerical fracture [JSS15]. In the Lagrangian energy case, we modify the approach of Jiang et al. [JSS15] to include the effects of plasticity and damage.

In MPM, the discrete state consists of a collection of particles that partition the domain based on initial volumes V_p^0, with time t^n positions $\mathbf{x}_p^n$ and with masses m_p computed from the initial mass density as $\rho(\mathbf{x}_p^0, t^0)V_p^0$ and linear and affine time velocities $\mathbf{v}_p^n$, $\mathbf{C}_p^n$ used for APIC particle/grid transfers [JSS15]. In the case of traditional particle-based MPM, each particle additionally stores the elastic portion of the deformation gradient $\mathbf{F}_p^{E,n}$ and yield surface size τ_{Cp}. In the case of mesh-based MPM, we assume there additionally exists a tetrahedron mesh connecting the particles $\mathbf{x}_p^n$. We use e to denote elements in the mesh and store $\mathbf{F}_e^{E,n}$ and τ_{Ce} per tetrahedron element, rather than per particle. Furthermore, in the mesh-based case, we must also store the plastic part of the deformation gradient $\mathbf{F}_e^{P,n}$.

An MPM time step from time t^n to t^{n+1} typically consists of three steps: (1) mass (m_p) and momentum $(m_p\mathbf{v}_p^n)$ are transferred from particles to the grid using weights $(w_{\mathbf{i}p}^n = N(\mathbf{x}_p^n - \mathbf{x}_\mathbf{i}))$ defined by Eularian grid interpolating functions $N(\mathbf{x})$ that describe the degree of interaction between particle p and grid node $\mathbf{i}$, (2) the grid momentum $(m_\mathbf{i}^n\mathbf{v}_\mathbf{i}^n)$ is then updated in a variational way from the potential energy in the system, and finally (3) the motion of the grid under the updated momentum is interpolated to the particles. In step (2), the discretization is done differently in the cases of standard particle-based MPM versus the mesh-based approach. The difference lies in how the deformation gradient is computed. In the case of standard particle-based MPM, the deformation gradient is stored per particle and is updated using an updated Lagrangian view. With this assumption the deformation gradient is computed as the product of the time t^n deformation gradient $\mathbf{F}_p^n$ and the deformation of the grid (evaluated at the particle) over the time step $\hat{\mathbf{F}}_p^{n+1} = (\mathbf{I} + \Delta t \sum_{\mathbf{i}} \mathbf{v}_{\mathbf{i}}^{n+1} \nabla w_{\mathbf{i}p}^n)$ where $\nabla w_{\mathbf{i}p}^n = \frac{\partial N}{\partial \mathbf{x}}(\mathbf{x}_p^n - \mathbf{x}_{\mathbf{i}})$ is the derivative of the grid interpolating functions. In the case of mesh-based elasticity, the deformation gradient is computed using mesh connectivity as in standard FEM [SB12, JSS15] $\mathbf{F}_e^{n+1} = \sum_p \mathbf{x}_p^{n+1} \nabla \tilde{N}_p(\mathbf{X}_e)$ where $\tilde{N}_p(\mathbf{X})$ is the piecewise linear interpolating function associated with particle p evaluated at the tetrahedron barycenter in the initial configuration of the mesh. We summarize this below as

$$m_{\mathbf{i}}^n = \sum_p w_{\mathbf{i}p}^n m_p \tag{3.15}$$

$$\mathbf{v}_{\mathbf{i}}^n = \frac{1}{m_{\mathbf{i}}^n} \sum_p w_{\mathbf{i}p}^n m_p (\mathbf{v}_p^n + \mathbf{C}_p^n (\mathbf{x}_{\mathbf{i}} - \mathbf{x}_p^n)) \tag{3.16}$$

$$\mathbf{v}_{\mathbf{i}}^{n+1} = \mathbf{v}_{\mathbf{i}}^n + \frac{dt}{m_{\mathbf{i}}^n} \mathbf{f}_{\mathbf{i}} + \Delta t \mathbf{g} \tag{3.17}$$

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \Delta t \sum_{\mathbf{i}} \mathbf{v}_{\mathbf{i}}^{n+1} w_{\mathbf{i}p}^n \tag{3.18}$$

$$\mathbf{v}_p^{n+1} = \sum_{\mathbf{i}} \mathbf{v}_{\mathbf{i}}^{n+1} w_{\mathbf{i}p}^n \tag{3.19}$$

$$\tilde{\mathbf{C}}_p^{n+1} = \frac{12}{\Delta x^2 (b+1)} \sum_{\mathbf{i}} w_{\mathbf{i}p}^n \mathbf{v}_{\mathbf{i}}^{n+1} \otimes (\mathbf{x}_{\mathbf{i}} - \mathbf{x}_p^n) \tag{3.20}$$

$$\mathbf{C}_p^{n+1} = (1-\nu)\, \tilde{\mathbf{C}}_p^{n+1} + \frac{\nu}{2} \left(\tilde{\mathbf{C}}_p^{n+1} - \tilde{\mathbf{C}}_p^{n+1T} \right) \tag{3.21}$$

$$\tilde{\mathbf{F}}_e^E = \left(\sum_p \mathbf{x}_p^{n+1} \nabla \tilde{N}_p(\mathbf{X}_e) \right) (\mathbf{F}_e^{P,n})^{-1} \tag{3.22}$$

$$\tilde{\mathbf{F}}_p^E = (\mathbf{I} + \Delta t \sum_{\mathbf{i}} \mathbf{v}_{\mathbf{i}}^{n+1} \nabla w_{\mathbf{i}p}^n) \mathbf{F}_p^{E,n} \tag{3.23}$$

$$\mathbf{F}_q^{E,n+1} = \text{returnMap}(\tilde{\mathbf{F}}_q^E). \tag{3.24}$$

Here the transfer to grid in step (1) consists of Equations (3.15)-(3.16), the grid-based momentum update in step (2) consists of Equations (3.17)-(3.19) and the interpolation from grid to particles in step (3) consists of Equations (3.19)-(3.21). This is using APIC transfers [JSS15] for Equations (3.16) and (3.20) as well as the RPIC damping of [JGT17] in Equation (3.21) where ν controls the amount of damping. Note that in Equation (3.17), $\alpha = 0$ corresponds to symplectic Euler for the grid momentum update and $\alpha = 1$ corresponds to backward Euler. Equations (3.22) and (3.23) represent the deformation gradient update in the cases of mesh-based and standard MPM respectively. Equation (3.24) projects the elastic state to satisfy the plasticity constraints. The equation is indexed by q to indicate that it is either e for mesh-based or p for particle-based MPM.

In Equation (3.17), $\mathbf{f}_\mathbf{i}$ is the force on grid node $\mathbf{i}$ which is computed as the variation of the total potential with respect to grid nodes moving as $\mathbf{x}_\mathbf{i} + \Delta t \mathbf{v}_\mathbf{i}^{n+\alpha}$, where $\alpha = 0$ corresponds to symplectic Euler and $\alpha = 1$ corresponds to backward Euler time stepping. The value varies based on the choice of mesh- or particle-based MPM as

$$\mathbf{f}_\mathbf{i} = \begin{cases} \sum_p w_{\mathbf{i}p}^n \mathbf{f}_p(\mathbf{x}^{n+\alpha}) + \Delta t \mathbf{g}, \\ -\sum_p \frac{\partial \psi}{\partial \mathbf{F}^E}(\tilde{\mathbf{F}}_p^E(\tilde{x}^{n+\alpha}))(\mathbf{F}_p^{E,n})^T \nabla w_{\mathbf{i}p}^n V_P^0 + \Delta t \mathbf{g} \end{cases} \tag{3.25}$$

respectively, where $\mathbf{x}^{n+\alpha} \in \mathbb{R}^{2n_P}$ is the vector consisting of all particle time $t^{n+\alpha}$ positions $\mathbf{x}_p^{n+\alpha}$ according to Equation (3.18). In the case of standard particle MPM, $\tilde{\mathbf{x}}^{n+\alpha}$ is the vector of all Eulerian grid node positions, moved according to

$$\mathbf{x}_\mathbf{i}^{n+\alpha} = \begin{cases} \mathbf{x}_\mathbf{i}, & \alpha = 0 \\ \mathbf{x}_\mathbf{i} + \Delta t \mathbf{v}_\mathbf{i}^{n+1}, & \alpha = 1 \end{cases} \tag{3.26}$$

In the case of mesh-based MPM, the particle force $\mathbf{f}_p$ in Equation (3.25) is related to the variation of the potential as estimated over the tetrahedron mesh, rather than the particles

$$\mathbf{f}_p = \sum_e \frac{\partial \psi}{\partial \mathbf{F}^E}(\tilde{\mathbf{F}}_e^E(\mathbf{x}^{n+\alpha}))\nabla \tilde{N}(\mathbf{X}_e) \tag{3.27}$$

where $\tilde{\mathbf{F}}_e^E(\mathbf{x}^{n+\alpha})$ is given by Equation (3.23).

3.5 Material surface definition and visualization

We provide a novel pre-scoring strategy for visualization of material boundary and crack surfaces as a post-process for ductile fracture simulations. Our approach can easily be used for most standalone MPM solvers. Our technique works with either traditional particle-based MPM, or Lagrangian energy mesh-based MPM [JSS15]. In the case of mesh-based MPM, we assume the user provides a tetrahedron mesh of quality suitable for FEM simulation of elasticity. In the case of traditional particle-based MPM, we assume the user provides interior points that are sampled with a Poisson disc, or similar initial random spacing. We also assume that the user provides a triangulation of the boundary of the domain from which the internal particles are sampled. The vertices of the boundary (triangle) mesh and the randomly sampled interior particles are treated as MPM particles for simulation. If the user does not provide a triangle mesh, we can generate one by surfacing the interior particles using an existing technique like [YT13]. We assume that most users will define the boundary of the initial domain for ductile materials using a triangle mesh, typically with texture etc. and our approach is designed to preserve those details throughout the simulation. Once in possession of the boundary triangle mesh and the interior particles, we create a Delaunay tetrahedralization connecting the interior and boundary points and preserving triangles on the original boundary.

3.5.1 Visualization mesh topology

With our initialization strategy, in either the traditional particle-based MPM or Lagrangian energy mesh-based MPM cases, we can assume we have a tetrahedralization of the particles used in the MPM calculation. The mesh is used to define a particle-wise partition of the material domain. Each tetrahedron in the mesh is split into four cuboids, one for each of

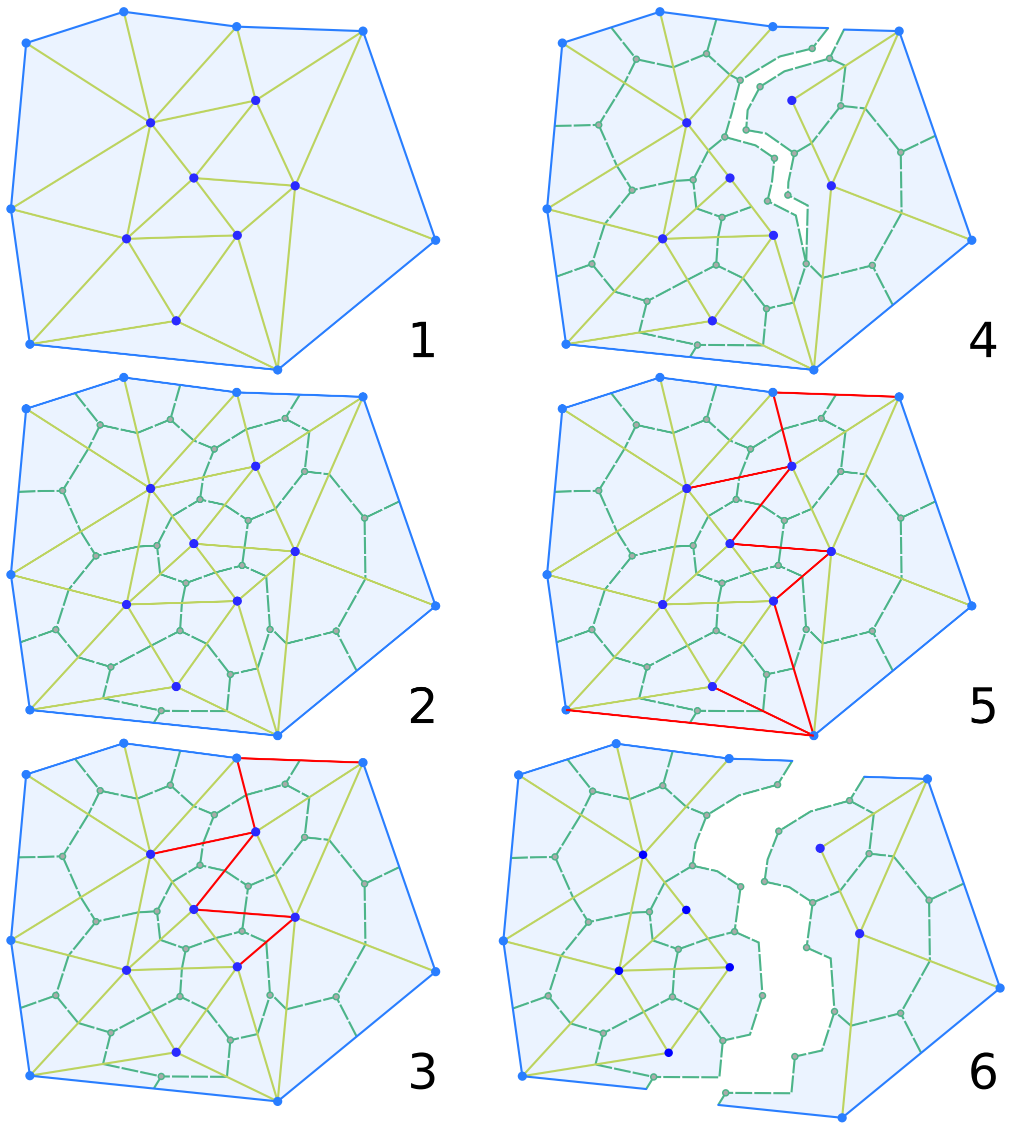

Figure 3.12: **Mesh cutting**. From left to right: 1: Initial simplex mesh (Delaunay or quality mesh generated for Lagrangian simulation). 2: Particle core partitioning. 3: Identify failed edges (marked red). 4: The corresponding partially split mesh to the set of failed edges in 3. 5: A different set of failed edges (marked red). 6: The corresponding split mesh to the set of failed edges in 5.

its particles. To create the particle-wise partitioning, each particle in the MPM calculation receives a cuboid from each of the tetrahedron elements it belongs to. We note that this is essentially the same as the per-particle cores of material used in the virtual node approach of Molino et al. [MBF04]. We adopt this name and refer to the particle's union of cuboids as its core of the domain. With this convention, each particle is responsible for updating its core over the course of the simulation.

The boundary of each particle core initially shares faces with cores of particles that it is connected to in the tetrahedron mesh. We define material failure on a per-initial-tetrahedron-mesh-edge basis. That is, common faces on cores of material associated with particles initially connected in the tetrahedron mesh are treated as identical until material failure occurs. To define material failure, we label core faces between particles connected along an edge in the tetrahedron mesh as broken. We use a simple union-find data structure to manage the topological connectivity and create a hexahedron mesh that respects the failed core faces. To do this we start with a mesh that is completely broken into the maximally split configuration and merge unbroken faces using the union-find data structure. See Figure 3.12 for details. One could use an element wise splitting strategy where core faces within a damaged element are broken, but we found that this gave inferior results to this edge-wise criterion.

We manage all topological aspects of the material and crack surface visualization with this simple strategy. Next we discuss our criteria for deciding when an edge (and its associated core faces) are broken as well as the geometric aspects of the crack surface evolution.

3.5.2 Topology evolution

We use a history-based maximal stretching criteria to define broken edges. We define the maximum relative stretching of an edge for times before a given time t as

$$\zeta_t = \max_{s<t} \frac{\|\phi(\mathbf{X}_1, t) - \phi(\mathbf{X}_2, t)\|}{\|\mathbf{X}_1 - \mathbf{X}_2\|}. \tag{3.28}$$

When this value is larger than a threshold, we consider the cores associated with $\mathbf{X}_1$ and $\mathbf{X}_2$ as separated from each other and break the edge connecting them. Note that if any edge is broken at a given time $\hat{t}$ it will be broken for all times $t > \hat{t}$.

3.5.3 Visualization mesh geometry: extrapolation

Each particle is responsible for updating the geometry of its core. We do this with a simple extrapolation strategy. We use a rigid transform local to each particle to extrapolate the motion of the particle to the rest of its core. For each core vertex $\mathbf{y}_p^n$ associated with a particle center $\mathbf{x}_p^n$, we compute the time t^n position as

$$\mathbf{y}_p^n = \mathbf{R}_p^n(\mathbf{y}_p^0 - \mathbf{x}_p^0) + \mathbf{x}_p^n, \tag{3.29}$$

where $\mathbf{R}_p^n$ is the rotation associated with the simulated particle p at time t^n. We use the MPM grid velocity to update the local rotation matrix on each particle

$$\mathbf{Z}_p^{n+1} = \left(\mathbf{I} + \sum_i \tilde{\mathbf{v}}_i^n \nabla \omega_{ip}^n\right) \mathbf{R}_p^n, \tag{3.30}$$

$$\mathbf{R}_p^{n+1}\mathbf{S}_p^{n+1} = \mathbf{Z}_p^{n+1}. \tag{3.31}$$

where the polar decomposition $(\mathbf{R}_p^{n+1})^T\mathbf{R}_p^{n+1} = \mathbf{I}$, $\mathbf{S}_p^{n+1} = (\mathbf{S}_p^{n+1})^T$ is used to enforce orthogonality. This creates a rigid core translating and rotating with the particle. However, when the vertices on the boundary of the core are associated with multiple cores, we take the average of the extrapolated positions given by each core. This introduces visually realistic deformation when material is not fully failed, while reverting to translation and rotation in

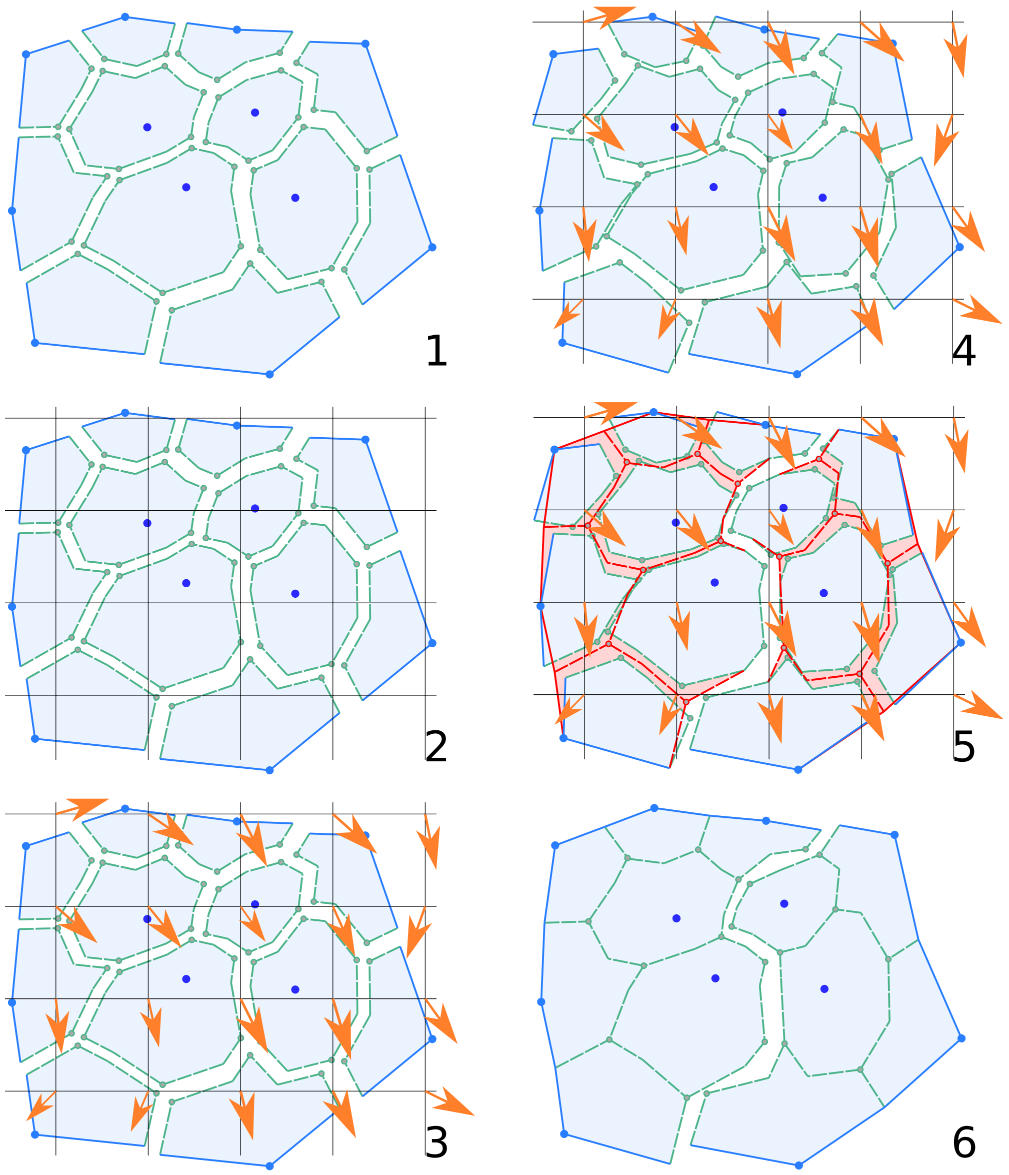

Figure 3.13: **Extrapolation**. 1. Initial particle core partition. 2. Initial particle core partition embedded in grid. 3. Velocity field defined on grid. 4. Particle cores positioned and oriented by local rigid body transform. 5. Sewing connected cells. 6. Final deformed fractured mesh.

the event of a fully separated core.

The accuracy of the update in Equation (3.30) is affected by the particle sampling density. If the grid resolution is too high relative to the particle density, the update can be noisy. For traditional particle-based MPM this is not an issue, however for Lagrangian energy MPM we found it advantageous to add traditional MPM particles in each element to help resolve update in Equation (3.30). These particles are not used to compute forces until their parent elements fail. In the event of failure, they function as standard elastic MPM particles. See Figure 3.15 on the right for details.

3.5.4 Visualization mesh geometry: crack smoothing

There is considerable flexibility when defining the initial geometry of each particle core. The geometry of the cuboid is most naturally chosen by setting its vertices as the edge, face and tetrahedron centers. However, these points may be chosen anywhere in their respective submanifolds. The only points on the cuboids without flexibility are those corresponding to MPM particles (tetrahedron mesh vertices). We take advantage of this flexibility to remove sampling based biasing in the crack paths. Note that the flexibility is only in the initial geometry of the cuboids. Once set, they must always evolve according to the per-particle extrapolation in Section §3.5.3.

A limitation of our pre-scoring visualization approach is that all possible crack paths are determined from the initial particle partitioning of the domain. This will lead to sampling bias of the crack surface in general. This tends to make the crack surfaces appear more jaggy in the case of randomly sampled initial points. In the case of structured initial points, the structure is imposed on the crack paths. In order to remove initial sampling bias, we iteratively smooth the crack surface in the initial configuration. Smoothing the surface tends to remove sampling bias as is usually visible through regions of locally high curvature. Because

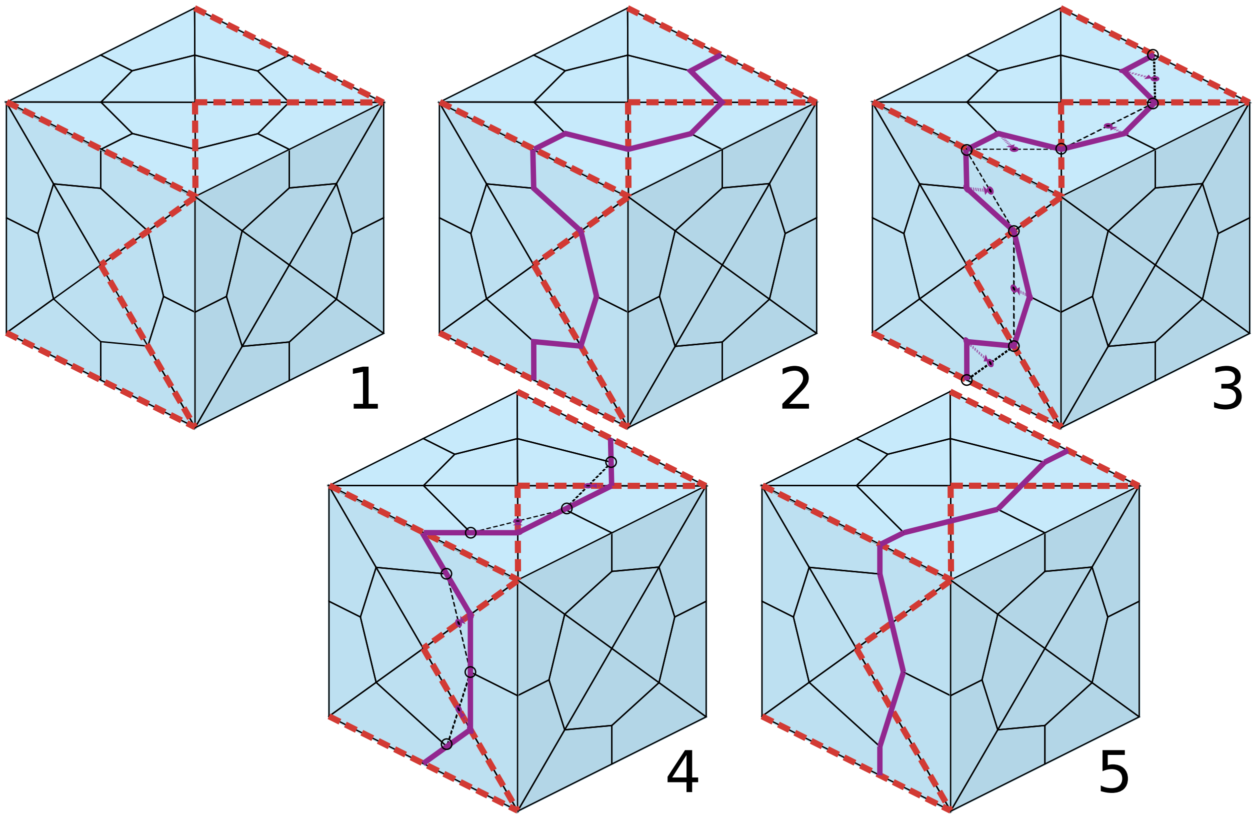

Figure 3.14: **Crack boundary curve smoothing**. From left to right: 1: Identify broken edges (red dashed line). 2: Identify boundary curve of the crack surface (purple solid line). 3-4: Smooth crack boundary curve while remaining on the original boundary surface: triangle centers move to average of neighbors, edge centers move to the intersection of its associated edge and the path joined by its neighbors. 5: Crack boundary curve after one iteration of smoothing.

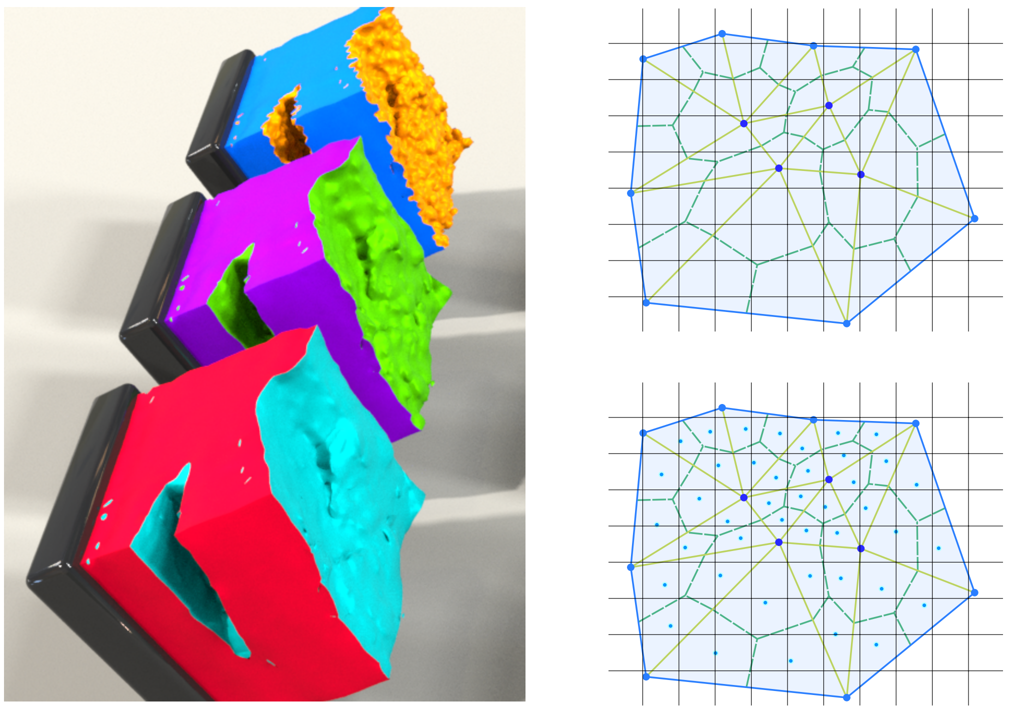

Figure 3.15: **Crack smoothing and sampling extra particles.** Left: original crack surface (yellow), crack surface smoothed with 2 iterations (green), crack surface smoothed with 20 iterations (cyan). Right: we sample extra particles in each quadrilateral/cuboid to help reduce noise.

our visualization technique is a post-process, we can assume that we know the topology of the crack surface at the final time from the condition in Section §3.5.2. We can therefore smooth the entire surface in the initial configuration, as required.

The first step of our approach smoothes the intersection of the initial material boundary surface and the crack surface. Care must be taken in this step to ensure that the boundary crack curves remain on the initial boundary during the smoothing process. See Figure 3.14 for details. Next, we smooth the crack surface interior by assigning each vertex to the average of its neighbors while the curve processed in the first step remains unchanged. We do this in a Gauss-Seidel fashion. Our approach quickly removes high-frequency noise while preserving the general shape of the crack pattern.

3.6 Results

We demonstrate our ductile fracture simulation and surface visualization techniques with a variety of simulations exhibiting a wide range of representative behaviors. We list our computational performance and simulation details in Table 3.1. We note that in many of our examples, remarkably detailed fracture patterns are produced with comparatively low resolutions. This is advantageous because surfacing limitations often require simulations with artificially high resolution in many MPM applications. Our results were run on an Intel Xeon E5-2687W v4 with 48 threads. Time stepping was adaptively chosen according to the CFL condition, i.e. Δt was set so no particle travels more than a portion of a grid cell in each time step. For particle-based MPM, the grid resolution was chosen so that there are initially approximately six particles per grid cell. For Lagrangian energy MPM, the grid resolution reflects the tetrahedron mesh resolution, i.e. grid Δx was chosen roughly the same as the average edge length of the tetrahedron mesh. In our examples, we used TetWild to generate the tetrahedron mesh for Lagrangian MPM [HZG18].

	Simulation	Post-process	Resolution
Pull - MPM (Fig. 3.17 red and blue)	0.6	0.5	8K
Pull - Lagrangian (Fig. 3.17 green)	0.6	0.5	8K
Projectile - 77K (Fig. 3.7 red)	2	5	77K
Projectile - 33K (Fig. 3.7 blue)	0.9	2	33K
Projectile - 14K (Fig. 3.7 yellow)	0.4	0.7	14K
Projectile - 5.5K (Fig. 3.7 orange)	0.2	0.3	5.5K
Twist - 60K (Fig. 3.6 blue)	11	5	60K
Twist - 17K (Fig. 3.6 purple)	4	1	17K
Twist - 8K (Fig. 3.6 green)	2	0.4	8K
Twist - 4K (Fig. 3.6 red)	2	0.2	4K
Twist von Mises (Fig. 3.10)	11	4	60K
Pulling with angle - 60K (Fig. 3.6 blue)	11	5	60K
Pulling with angle - 17K (Fig. 3.6 purple)	8	1	17K
Pulling with angle - 8K (Fig. 3.6 green)	8	0.4	8K
Pulling with angle - 4K (Fig. 3.6 red)	5	0.2	4K
Braiding Columns (Fig. 3.4 and Fig. 3.5)	35	16	200K
Crushing Orange (Fig. 3.3)	15	8	130K
Zucchini (Fig. 3.1 bottom)	16	13	207K
Stretching Armadillo (Fig. 3.8)	49	27	299K
Tearing Armadillo (Fig. 3.9)	48	26	299K
Wall breaking (Fig. 3.1 right)	50	5	933K

Table 3.1: Performance of Ductile Fracture Simulations

All simulations and post-processes were run on an Intel Xeon E5-2687W v4 with 48 threads and 128 GB of RAM. Simulation and post-process time are measured in averaged seconds per frame, and resolution is measured by particle count.

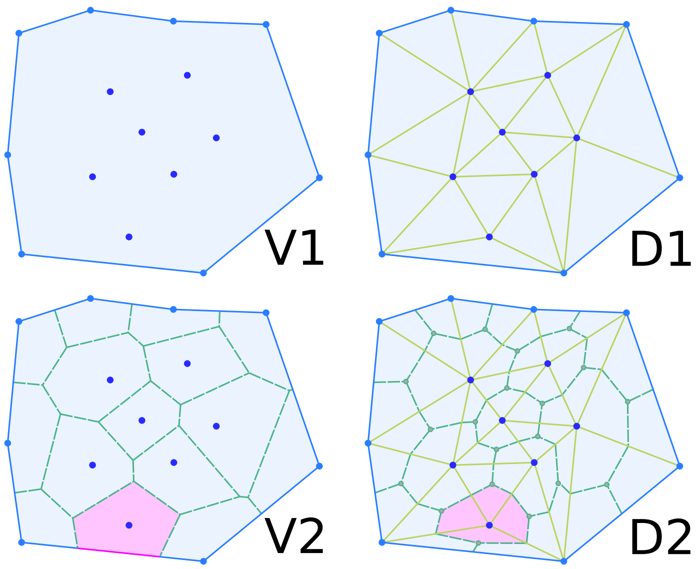

Figure 3.16: **Voronoi versus Delaunay** Given a point cloud and a boundary surface (V1), its Voronoi diagram could be ill-posed where interior cell intersects the boundary (V2). If we take the dual of the Voronoi diagram, its Delaunay triangulation (D1), we can construct the degenerated Voronoi region (D2) without interior cell contacting the boundary.

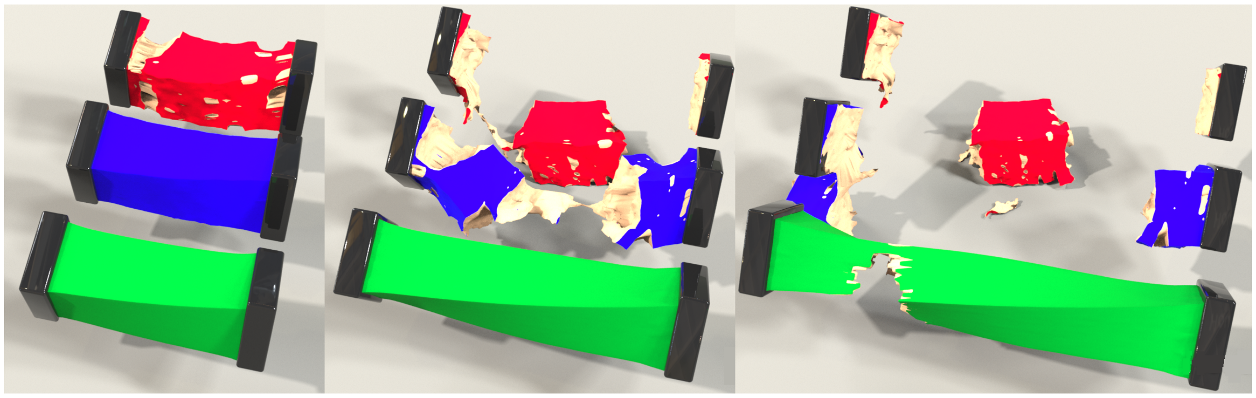

Figure 3.17: **Comparison of particle-based MPM and Lagrangian MPM.** We illustrate our treatment of numerical fracture with three simulations using the same particles. The red cube and blue cubes are simulated using traditional particle-based MPM with fine grid resolution (approximately 1 particle per grid cell) and coarse grid resolution (approximately 6 particles per grid cell) respectively. The green cube is simulated with our Lagrangian approach and fine grid resolution.

3.6.1 Capturing different fracture modes

We test our method with fracture simulations in which excessive tension or shear force is applied. In Figure 3.17, we simulate the process of pulling on a cube and demonstrate how Lagrangian MPM prevents numerical fracture caused by excessive deformation. In Figure 3.6, we twist and pull a cube until the shearing forces cause material failure and the material becomes disconnected. In Figure 3.9, we pull the 4 limbs of the armadillo until they break and observe how the fracture introduces momentum to the torso. In Figure 3.10, we added the von Mises plasticity model to the particles to capture more shear-induced plastic deformation.

3.6.2 Texturing objects

Our mesh visualization technique has the advantage that it naturally accommodates texturing based on an input mesh. E.g. all particles from the initial mesh are in the cut mesh and it is trivial to obtain a consistent vertex ordering based on the initial mesh for simplified texturing. In Figure 3.1, we simulated a zucchini being broken in half and demonstrated that its detailed texture is preserved. Also in Figure 3.1, we textured the ductile walls broken by the walking mannequin with SCA logos. In Figure 3.3, we textured the ductile sphere and created convincing details in the fracture scene.

3.6.3 Relaxed resolution requirements

In Figure 3.2, we simulated twisting of a cube with 8,000 particles. We compared two different renders: conventional particle fluid surface reconstruction and our approach. Our result captures significantly more detail and does not suffer from reconnection due to proximity. We also provide similar resolution comparison in Figure 3.6, and Figure 3.7. With our meshing technique, the results still look comparable even with comparatively low resolution.

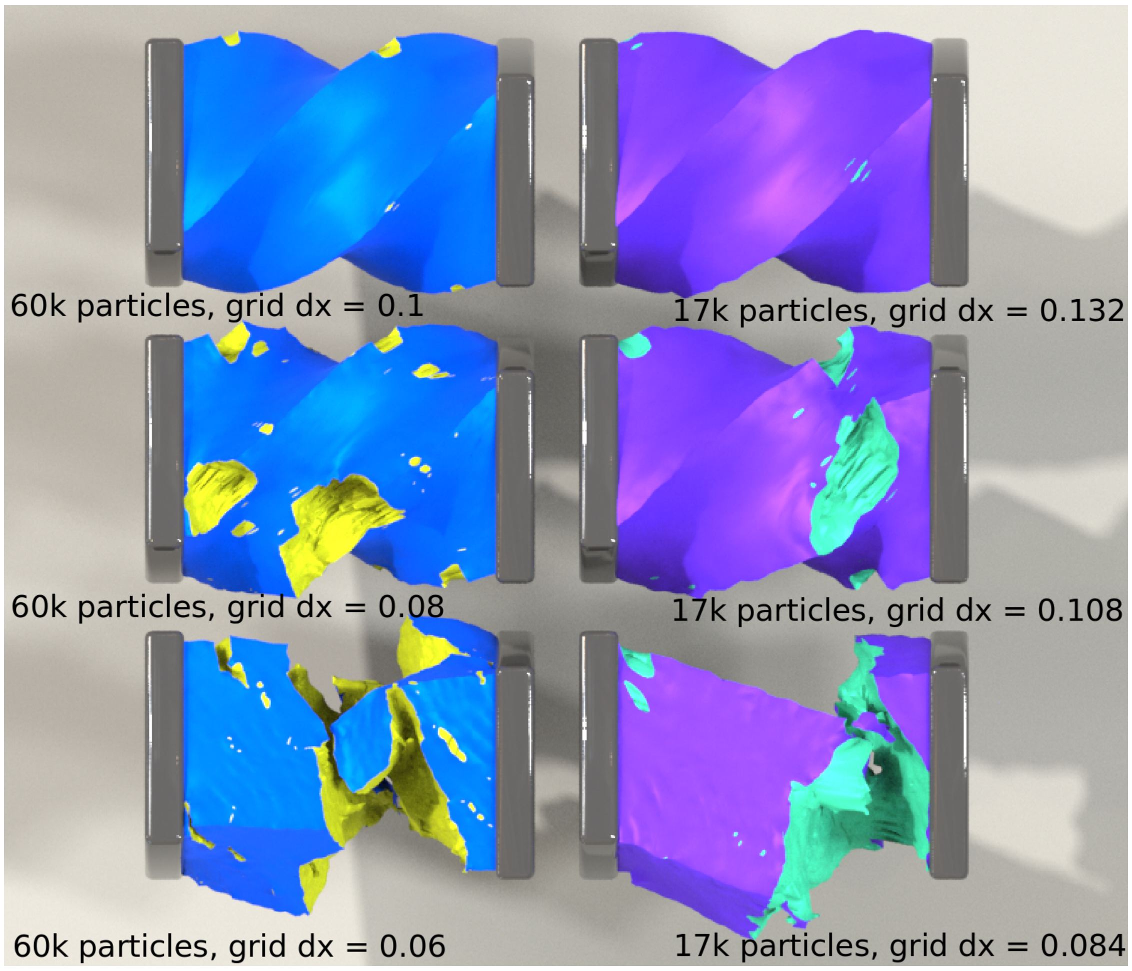
60k particles, grid dx = 0.1
17k particles, grid dx = 0.132
60k particles, grid dx = 0.08
17k particles, grid dx = 0.108
60k particles, grid dx = 0.06
17k particles, grid dx = 0.084

Figure 3.18: **Effect of grid resolution on fracturing behavior.** We compare the same twisting cube simulation with different particle count and grid size. The sims with smaller grid dx to particle count ratio experience more fracture than the ones with larger ratio in the same frame.

3.7 Discussion and limitations

Many existing FEM approaches for simulating ductile materials rely on the creation of a sufficiently high quality tetrahedron mesh to be used in the simulation. In the case of traditional particle based MPM, our mesh quality demands are practically non-existent. Indeed we simply use Delaunay tetrahedralization. In the case of Lagrangian mesh-based MPM our approach requires a mesh with the same quality constraints as traditional FEM. In either case, the MPM conception of our approach automatically resolves self-collision allowing us to simulate ductile fracture with comparably low implementation and computational complexity. Our approach does have a number of clear limitations. First, crack patterns are affected by particle sampling density/tetrahedron mesh topology and grid resolution. See Figure 3.18. Also, choosing appropriate parameters for edge splitting thresholds and crack surface smoothing iteration counts can vary from example to example.

3.8 Applications in visualizing thermomechanical simulations of baking and cooking

In [DHW19], we proposed an MPM-based simulation method for baking bread, cookies, pancakes, and similar materials that consist of dough or batter (mixtures of water, flour, eggs, fat, sugar, and leavening agents). We used a novel thermomechanical model using mixture theory to resolve interactions between individual water, gas, and dough species. Heat transfer with thermal expansion is used to model thermal variations in material properties. Water-based mass transfer is resolved through the porous mixture, gas represents carbon dioxide produced by leavening agents in the baking process and dough is modeled as a viscoelastoplastic solid to represent its varied and complex rheological properties. Water content in the mixture reduces during the baking process according to Fick's Law which contributes to drying and cracking of crust at the material boundary. Carbon dioxide gas

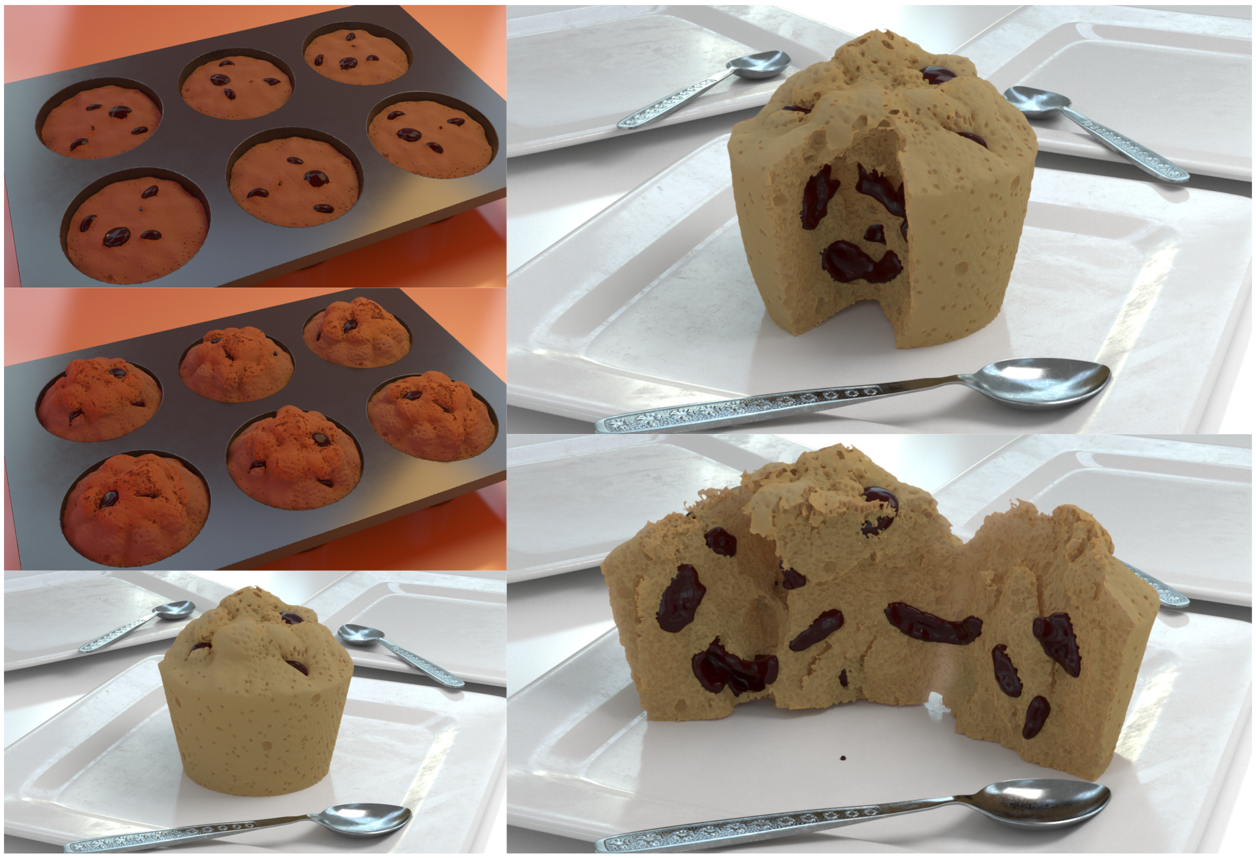
Figure 3.19: **Muffin.** Left: baking of a tray of muffins, resulting in a classic dome shape on top. Right: a muffin cut and torn open to reveal a fully cooked interior and melted chocolate chips. Surfacing the muffin particles would result in a blurry finish. Our mesh-processing technique helped show the crusty surface of the fractured muffin.

produced by leavening agents during baking creates internal pressure that causes rising. The viscoelastoplastic model for the dough is temperature dependent and is used to model melting and solidification.

These simulations are particle-based from their MPM conception. However, for simulations with fracture, we construct a reference tetrahedron mesh in the initial state for rendering purposes and adopt the post-processing techniques from [WDG19b] to obtain clean and consistent surfacing of the fractured material. The reference meshes are generated with

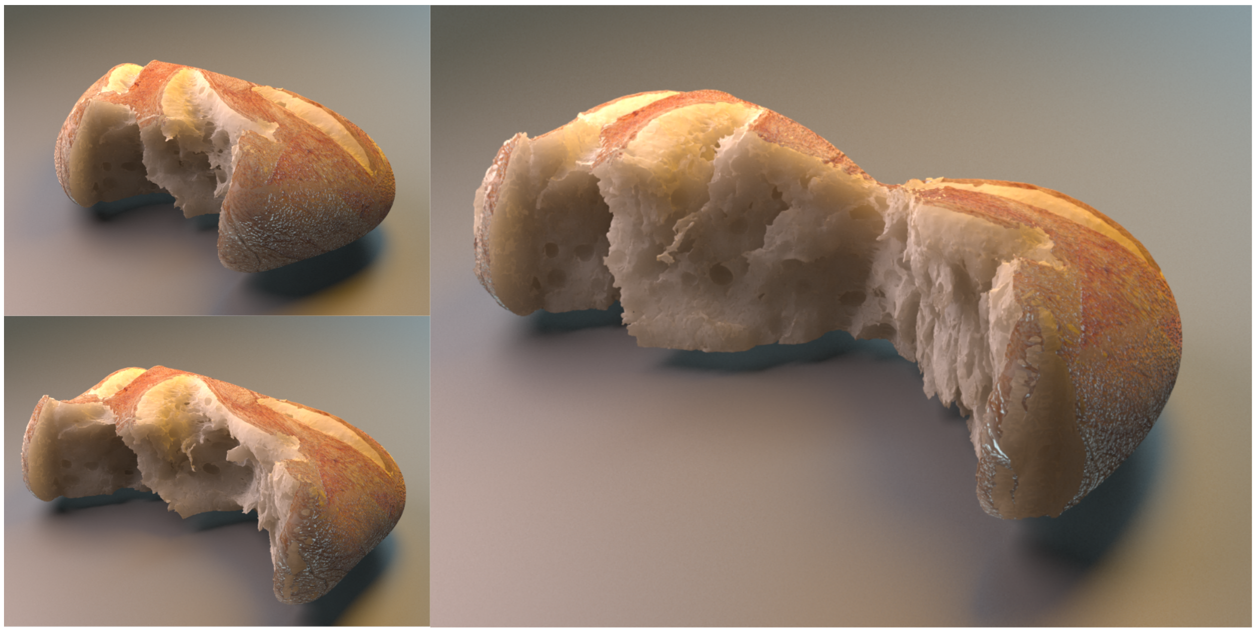

Figure 3.20: **Tearing bread**.

TetWild [HZG18]. We demonstrate crusty exterior and fibrous interior of the baked results with tearing examples in Figure 3.19 and Figure 3.20. By modeling the combined effect of water diffusion, temperature change, and chemical leavening, our method can achieve visually realistic baking and tearing of a muffin, see Figure 3.19. Drawing slits on the bread dough helps with the rising during baking as well as the formation of a nice crust. In Figure 3.21, we compare the baking process of bread with and without scoring the surface beforehand. Notice how the bread cracks in a more controlled and appealing manner when there are slits on the surface.

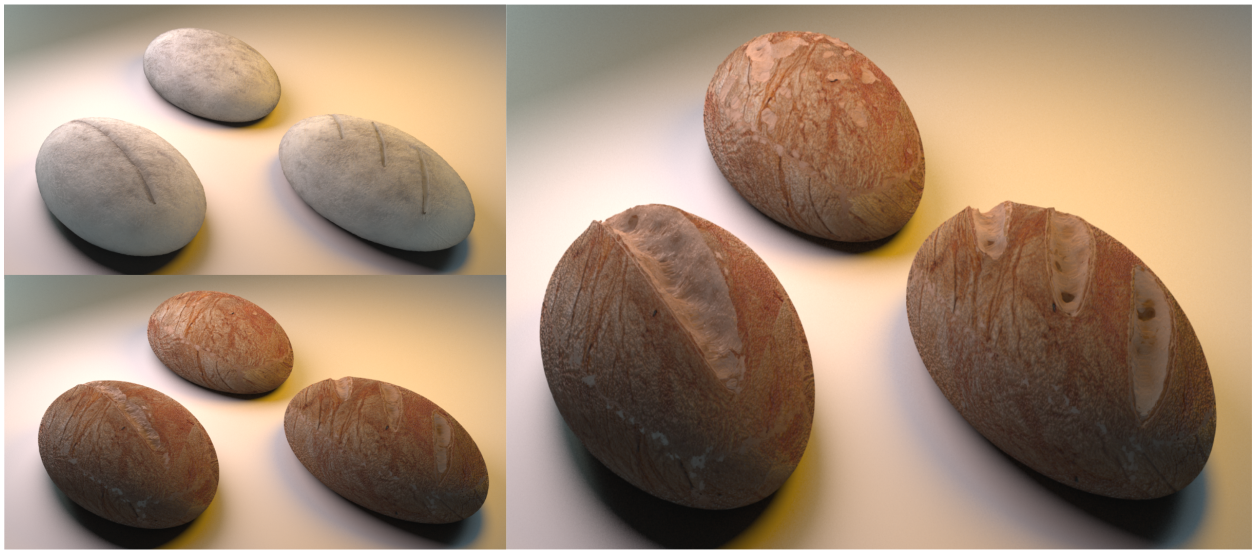

Figure 3.21: **Bread.** Top left shows raw dough, one is left intact and the other two have different slits on top. When baked (right), the bread expands in size and the slits open up. The bread without an initial slit also cracked on the top surface.

CHAPTER 4

Hybrid Material Point Method for Frictional Contact with Diverse Materials

4.1 Introduction

The Material Point Method (MPM) [SCS94] was developed as a generalization of the Particle-In-Cell (PIC/FLIP) [Har64, BR86] method to elastoplastic materials, and like PIC/FLIP, it has proven to be a very effective tool for many computer graphics problems. Phenomena like fracture/topological change, multiple material interactions, and challenging self contact scenarios with complex geometric domains are all commonplace in computer graphics applications. MPM naturally handles many of these. This was first demonstrated for snow dynamics by Stomakhin et al. [SSC13]. Since then a wide variety of other phenomena, particularly those that can be described as elastoplastic, have been simulated with MPM in graphics applications. This includes the dynamics of non-Newtonian fluids and foams [YSB15, RGJ15], melting [SSJ14, GTJ17], porous media [TGK17, GPH18, FBG18], and frictional contact between granular materials [DB16, KGP16, YSC18]. MPM has also been used to simulate contact and collision with volumetric elastic objects [JSS15, ZZL17] and frictional contact between thin hyperelastic materials like clothing and hair [JGT17, GHF18, FBG18]. In this paper, we refer to methods that follows Sulsky et al.'s original idea to use the updated Lagrangian view and grid interpolation functions to compute deformation as traditional MPM.

However, there are drawbacks associated with MPM collision resolution. As noted in

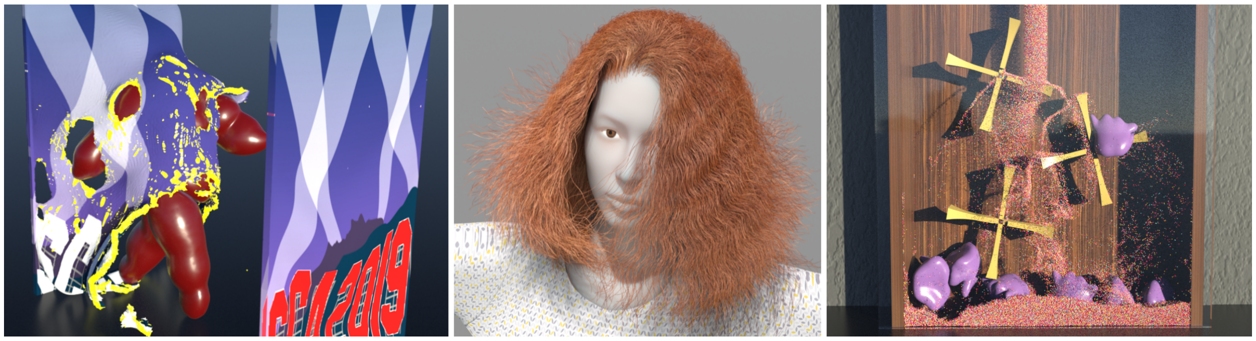

Figure 4.1: **Montage.** Left: Simulation of a mannequin breaking through an elastic wall. Middle: Hair of a dancer in motion. Right: Colored sand and elastic characters are poured into a cabinet, setting rigid pinwheels in motion.

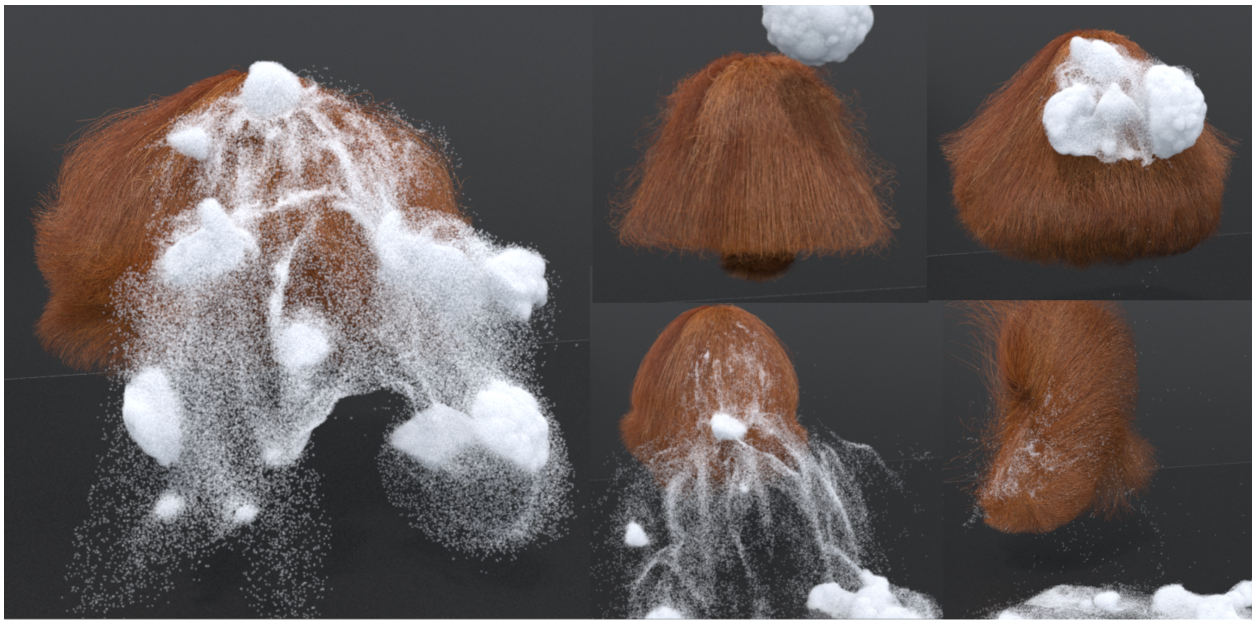

Figure 4.2: **Coupling hair with snow.** Our method captures the dynamics of a snowball falling on a head of hair.

[JSS15, FGG17, HN17], information is typically lost when transferring from particles to grid, since there are generally many more particles than grid nodes. Even when utilizing Lagrangian meshes in the updated Lagrangian view as in [JSS15, JGT17, GHF18, ZZL17] information is still lost which can lead to persistent wrinkles and apparent interaction at a distance, as discussed in [JGT17, GHF18]. Volumetric elastic materials suffer from two additional drawbacks. First, while contact for materials such as grains [KGP16, DB16], membranes/shells and fibers [JGT17, GHF18] can be envisioned as a continuum process where elastoplasticity associated with frictional contact is defined by the directions orthogonal to the grain, curve or surface, volumetric objects have no non-elastic directions for which to apply the condition. Hence, all self-collision resolution will result from volumetric elasticity, which means that frictional sliding cannot be regulated in a Coulomb fashion via plasticity. The second drawback is that the Eulerian grid spacing must be approximately the same as the edge lengths in the volumetric Lagrangian mesh. If the Eulerian grid resolution is significantly lower, there is non-negligible information loss in the transfer from particles to grid, and there will be spurious interaction at a distance. If the grid resolution is significantly higher, collisions will not be resolved (see Figure 4.7). This is problematic because visual separation between elastic bodies is proportionate to the Eulerian grid spacing, which therefore mandates high spatial resolution of the volumetric Lagrangian mesh to reduce separation thickness. This problem is not present when simulating cloth and hair because they admit the use of elastoplasticity frictional contact particles [JGT17, GHF18] and arbitrarily many can be added on each surface element or hair segment to accommodate high spatial grid resolution.

Our novel hybrid Lagrangian Material Point Method is designed to alleviate these drawbacks. Our approach utilizes more of the Lagrangian degrees of freedom to minimize artifacts while retaining aspects of MPM that allow for collision resolution without suffering from information loss when going from particles to grid. Our approach also resolves the Eulerian grid size (and artificial separation distance) limitations associated with volumetric elasticity, allowing for Coulomb frictional contact with volumetric elastic meshes. We support coupling with materials simulated with standard MPM discretizations and we provide for simple two-way coupling with rigid bodies. We demonstrate the effectiveness of our techniques with skinning, clothing, hair and multi-material simulation examples. In summary, our contributions are:

- Novel collision impulses defined from the MPM particle to grid transfers that resolve the drawbacks of the volumetric approaches in [JSS15, ZZL17].
- A hybrid elastoplastic model for hair and strand self collision that supports bending, torsion and stretching resistance and that does not suffer from information loss in particle to grid transfers.
- Two-way coupling with rigid bodies.
- Removal of numerical cohesion between phases.
- Coupling with materials discretized with traditional MPM.

4.2 Previous work

Our method fits most naturally within the context of PIC/MPM methods, but also with hybrid approaches and those that make use of Lagrangian and Eulerian techniques for self collision. Here we discuss the relevant computer graphics techniques within these categories.

McAdams et al. [MSW09] use a hybrid PIC/geometric impulse technique to resolve self collision of many thin straight hairs. They assume that hair is incompressible and interpret the PIC grid projection as a Lagrangian repulsion. They then apply the collision impulses of Bridson et al. [BFA02] to catch cases not resolved on the grid. Yue et al. [YSC18] develop a hybrid MPM/discrete element (DEM) technique. The DEM approach resolves frictional contact directly through constrained optimization and is generally much more detailed, but

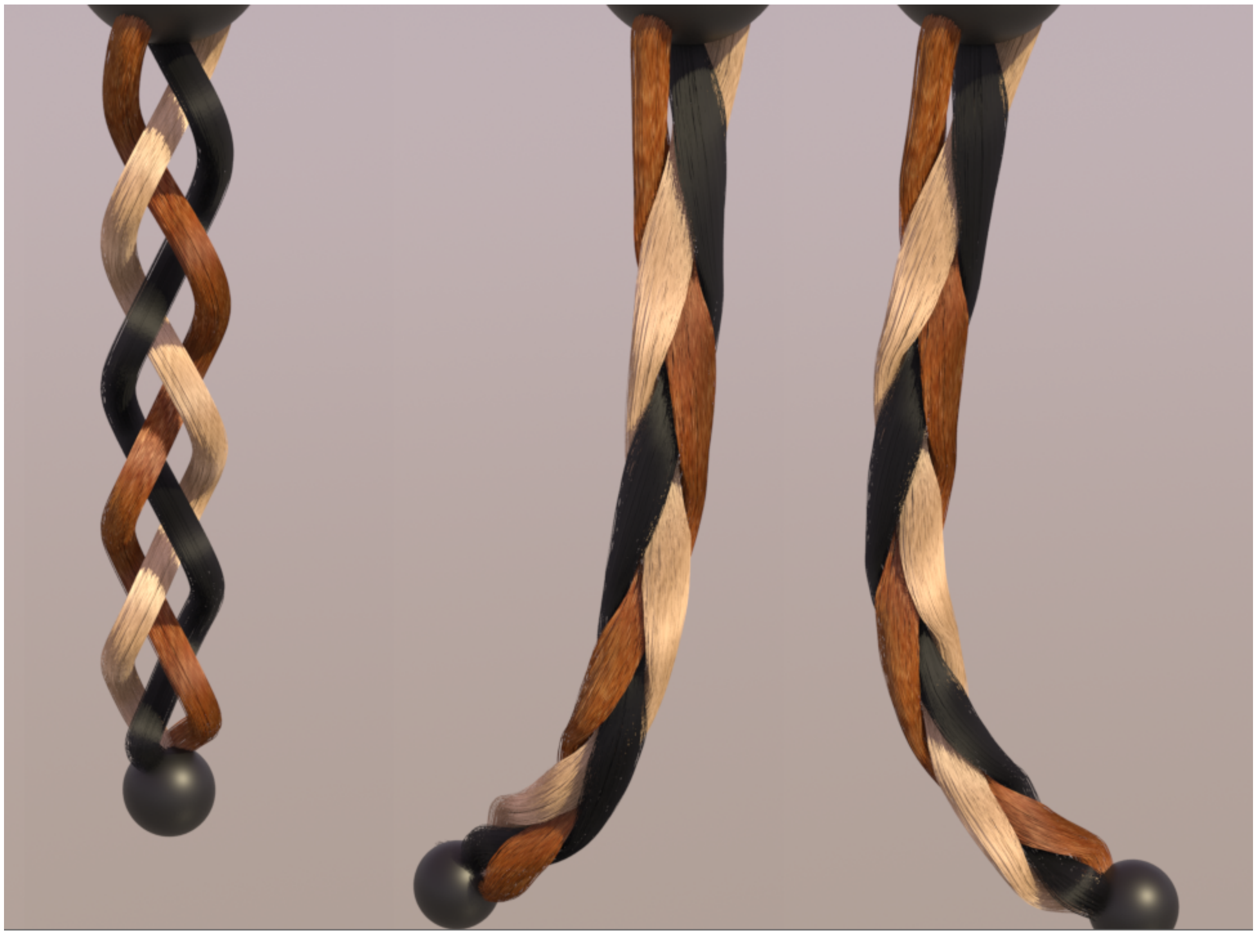

Figure 4.3: **Hair braids**. Our method captures the dynamics of a braid by robustly resolving many collisions.

more expensive. MPM is used where the expense of DEM would be prohibitive, and their technique resolves the combination of these two representations. Sifakis et al. [SSI07] also use multiple representations of elastic materials to help resolve contact, including the use of a high-resolution surface mesh to aid in collision resolution.

Pai and colleagues [LLJ11, FLL13, FLP14] pioneered a class of methods using Eulerian techniques for self collision with elastic objects. Li et al. [LSN13] show that the Eulerian view is useful for resolving close self contact between skin and other soft tissues. Teng et al. [TLK16] show that the approach can be naturally used to couple with incompressible fluids. Hybrid Eulerian/Lagrangian techniques are also useful for simulating crowd dynamics [NGC09, GNL14]. Our method is also similar to those of Müller et al [MCK15], Sifakis et al. [SMT08] and Wu et al. [WY16]. These approaches mesh the space surrounding elastic objects and enforce positive volume and/or incompressibility constraints respectively on the air surrounding the objects to resolve collisions.

MPM techniques have proven very effective in graphics applications. Stomakhin et al. [SSC13] and Gaume et al. [GGT18] use the method to simulate snow. Various others have simulated more general granular materials like sand [DB16, KGP16], porous water and sand mixtures [TGK17, GPH18], viscoelastic foams and sponges [YSB15, RGJ15], coupling with rigid bodies and cutting [HFG18], volumetric elastic materials [JSS15, ZZL17], thin elastic membranes and shells [JGT17, GHF18], and even wet clothing [FBG18]. Various improvements to the method have been made, including removal of noise with angular momentum conservation [JSS15, FGG17], adaptive spatial discretization [GTJ17], temporally asynchronous time stepping [FHH18], and GPU acceleration [GWK18]. Also of relevance is the approach of Huang et al. [HZM11] to N-body collision, which has been used for self collision for fracture debris in graphical simulation of ductile fracture by Hegemann et al. [HJS13].

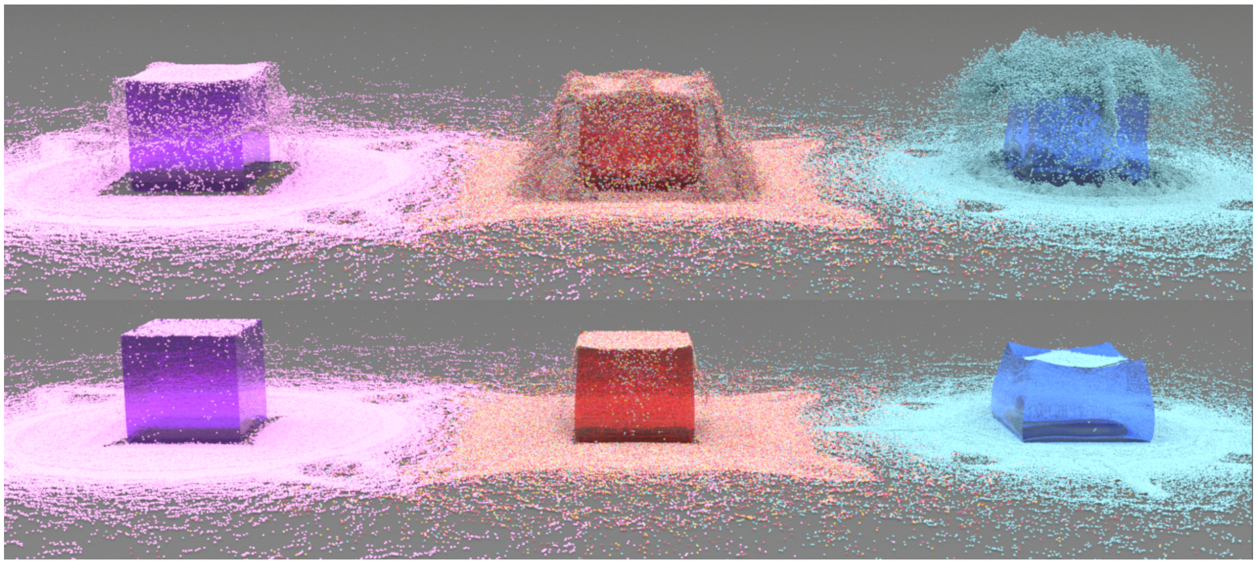

Figure 4.4: **MPM particle coupling**. Elastic Jell-O's with varying stiffness are two-way coupled with MPM particles.

4.3 Mathematical background

Here we describe the governing equations for volumetric elastic solids and hair strands. We define the deformation of an elastic body as a map from its undeformed configuration consisting of points $\mathbf{X}$ to its deformed configuration consisting of points $\mathbf{x}$ at time t by $\mathbf{x}(t) = \boldsymbol{\phi}(\mathbf{X}, t)$. We refer to the spatial derivative of this map as the deformation gradient $\mathbf{F} = \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}}$. The deformation gradient is used as a measure of strain, where its deviation from orthogonality indicates the local violation of rigid body motion. For hair, we decompose the deformation gradient into elastic and plastic parts $\mathbf{F} = \mathbf{F}^E\mathbf{F}^P$, where $\mathbf{F}^E$ is the elastic deformation and $\mathbf{F}^P$ is the plastic deformation, as a means to resolve stress constraints associated with frictional contact as in [KGP16, JGT17, GHF18]. For elastic solids, we do not use an elastoplastic decomposition. Instead, we model elastic objects using hyperelasticity [BW08], where the potential energy in the system increases as $\boldsymbol{\phi}$ deviates from rigid body motion. For frictional collision with hair strands, the potential energy density penalizes $\mathbf{F}^E$. We adopt the fixed corotational model from [SHS12] for elastic solids, the Discrete Elastic

Rod (DER) model from [BWR08, BAV10] for hair and strands, and the St. Venant-Kirchhoff Hencky model from [KGP16] for hair collision resistance.

The governing equations for the material deformation $\boldsymbol{\phi}$ are described from conservation of mass and momentum

$$\frac{D\rho}{Dt} + \rho\boldsymbol{\nabla}\cdot\mathbf{v} = 0,\ \ \rho\frac{D\mathbf{v}}{Dt} = \boldsymbol{\nabla}\cdot\boldsymbol{\sigma} + \rho\mathbf{g} \tag{4.1}$$

where

$$\boldsymbol{\sigma} = \frac{1}{J}\mathbf{P}\mathbf{F}^{E^T},\ \ \mathbf{P} = \frac{\partial\psi}{\partial\mathbf{F}^E},\ \ J = \det(\mathbf{F}). \tag{4.2}$$

ρ is material density, $\mathbf{v}$ is velocity, $\mathbf{g}$ is gravity constant, $\mathbf{P}$ is the first Piola-Kirchhoff stress, and $\boldsymbol{\sigma}$ is the Cauchy stress. ψ is the potential energy density, which we assume varies with $\mathbf{F}^E$. For volumetric objects we do not use an elastoplastic decomposition and can so assume $\mathbf{F}^E = \mathbf{F}$ in this case.

4.3.1 Hyperelastic volumetric solids

For volumetric elastic objects, we adopt the fixed corotational model from [SHS12], though any hyperelastic potential may be used. With this choice, the stress satisfies

$$\begin{gathered}\psi(\mathbf{F}) = \mu\sum_i(\sigma_i - 1)^2 + \frac{\lambda}{2}(J-1)^2,\\ \mathbf{P} = \mu(\mathbf{F}-\mathbf{R}) + \lambda(J-1)J\mathbf{F}^{-T}.\end{gathered} \tag{4.3}$$

Here μ and λ are the Lamé coefficients that express the material resistance for deformation and volume change, and σ_i are the singular values of the deformation gradient $\mathbf{F}$ computed according to the polar SVD convention of [ITF04] to allow for extreme deformation.

4.3.2 Hair strands

We follow the codimensional approaches of [JGT17, GHF18] and penalize frictional contact between hairs and thin strands using a continuum assumption. Following their formulation,

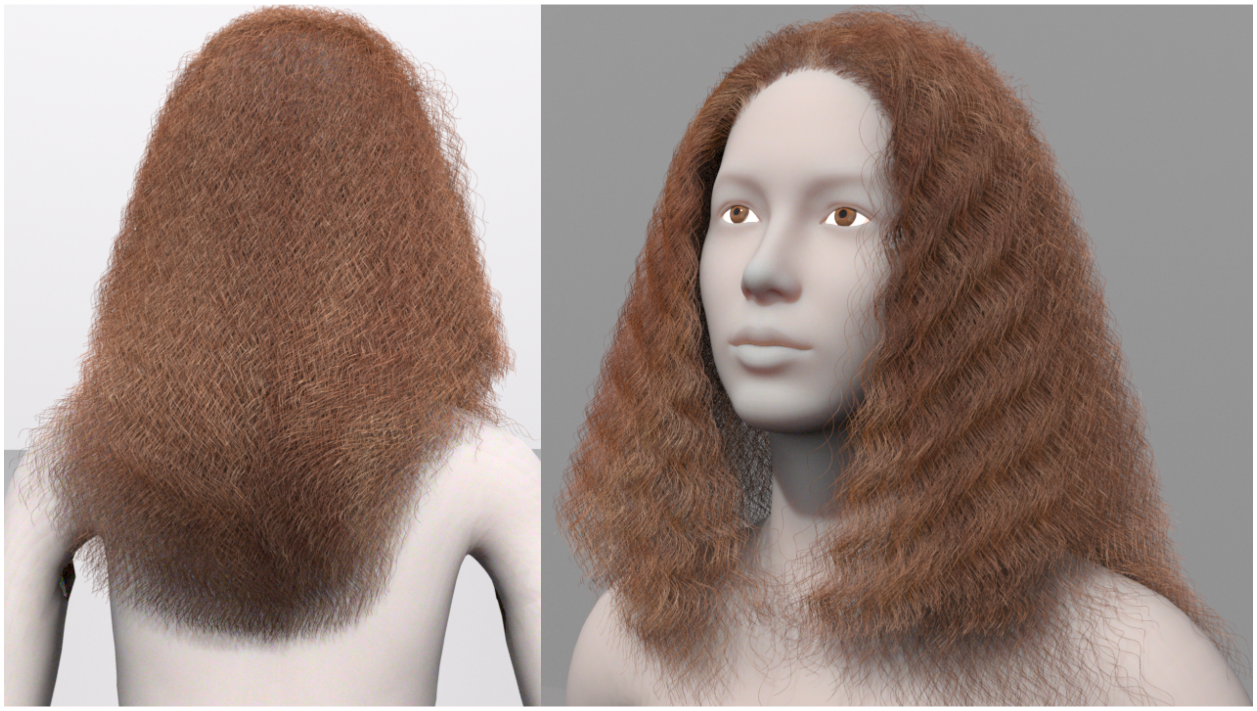

Figure 4.5: **A walking mannequin with a full head of hair**.

we decompose the deformation of the material $\boldsymbol{\phi}$ into the deformation of the individual strands $\boldsymbol{\phi}^s$ and the deformation associated with frictional contact interactions among strands $\boldsymbol{\phi}^d$, namely

$$\boldsymbol{\phi} = \boldsymbol{\phi}^d \circ \boldsymbol{\phi}^s. \tag{4.4}$$

Consequently, the deformation gradient is decomposed into $\mathbf{F} = \mathbf{F}^d\mathbf{F}^s$. We treat the deformation of the strand $\mathbf{F}^s$ as purely elastic using standard rod and curve models [BWR08, BAV10, BAC06, MSW09], and decompose $\mathbf{F}^d$ into elastic and plastic components,

$$\mathbf{F}^d = \mathbf{F}^{d,E}\mathbf{F}^{d,P} \tag{4.5}$$

to handle frictional contact among hair strands.

We utilize the continuum Coulomb friction view from [KGP16, JGT17, GHF18] to place a constraint on admissible stress. Shear stresses resisting sliding motions between strands cannot be larger than a frictional constant times the normal stress holding them together.

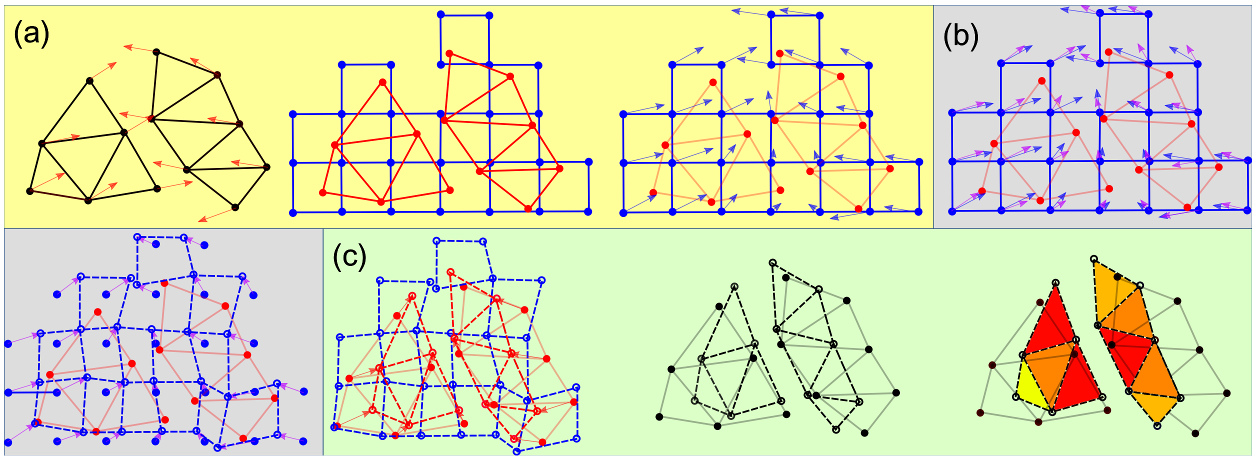

Figure 4.6: **MPM Overview**. The steps in the MPM update are: (a) The Lagrangian quantities (black and red) are transferred to an Eulerian grid (blue), which may be viewed as a new FEM mesh. (b) Grid nodes receive new velocities (purple) from updated Lagrangian elastic updates and are temporarily moved with those velocities. (c) The Lagrangian quantities are updated by interpolating from the new positions and velocities of the Eulerian grid nodes. The triangles are colored based on the amount of compression.

When the shear stress exceeds that threshold, the strands will start to slide against each other, inducing plastic deformation. Mathematically, the Coulomb friction model states that $\mathbf{s}^T\boldsymbol{\sigma}\mathbf{n}+c_F\mathbf{n}^T\boldsymbol{\sigma}\mathbf{n} \leq 0$, where $\mathbf{n}$ is the normal to the contact surface, $\mathbf{s}$ is any unit vector along the contact surface, and c_F is the friction coefficient. While Jiang et al. [JGT17] considers only directions $\mathbf{n}$ orthogonal to the tangent of the midline of the strand, we enforce this condition for all directions. The continuum assumption in Jiang et al. [JGT17] is that of a tube of parallel strands, which holds well for simulating knits but is less effective in the more complicated contact scenarios that occur when simulating hair and thin strands. To accomodate this more general constraint, we use an isotropic potential to resist collision, rather than the transversely isotropic potential of Jiang et al. [JGT17].

With this convention, we define the potential energy as a combination of the DER energy for strand elasticity and the St. Venant-Kirchhoff Hencky energy from [KGP16] to penalize collision and shearing,

$$\Psi = \Psi^s(\mathbf{F}^{d,E}) + \Psi^{DER}(\mathbf{F}^s). \tag{4.6}$$

The St.Venant-Kirchhoff Hencky energy, chosen for the ease of plasticity return mapping, takes the form

$$\Psi^s = \int_\Omega \psi^s dV \tag{4.7}$$

$$\psi^s = \mu \mathrm{tr}\left((\ln \boldsymbol{\Sigma})^2\right) + \frac{1}{2}\lambda\left(\mathrm{tr}\,(\ln \boldsymbol{\Sigma})^2\right) \tag{4.8}$$

where $\mathbf{F}^{d,E} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$ is the singular value decomposition of the elastic deformation, Ω is the original domain the material occupies, and μ and λ are Lamé parameters. The DER energy Ψ^{DER} consists of stretching, twisting, and bending potentials. We refer readers to [BAV10] for details on this energy and the time parallel transport required to calculate the force. The derivatives of the potential with respect to deformation are needed for computation and satisfy

$$\frac{\partial \psi^S}{\partial \mathbf{F}^E}(\mathbf{F}^E) = \mathbf{U}\left(2\mu\boldsymbol{\Sigma}^{-1}\ln(\boldsymbol{\Sigma}) + \lambda\boldsymbol{\Sigma}^{-1}\ln(\boldsymbol{\Sigma})\right)\mathbf{V}^T. \tag{4.9}$$

4.4 Discretization: hyperelastic solids

Our hybrid approach utilizes aspects of traditional Finite Element Methods (FEM) for hyperelasticity [SB12]. However, our approach is largely motivated by the the MPM treatment of volumetric objects from Jiang et al. [JSS15] and Zhu et al. [ZZL17]. These methods were originally designed to prevent the numerical fracture that would occur with volumetric objects in traditional particle-based MPM. We first discuss this approach and how it resolves self collision, followed by its drawbacks.

In Jiang et al. [JSS15] and Zhu et al. [ZZL17], the state at time t^n consists of particles with positions $\mathbf{x}_p^n$ connected with a tetrahedron mesh with elements indexed by e, as in Lagrangian FEM. Furthermore, particles store velocities $\mathbf{v}_p^n$ and masses m_p. The MPM time step from time t^n to t^{n+1} consists of three steps: (1) mass (m_p) and momentum ($m_p\mathbf{v}_p^n$) are transferred from particles to the grid using weights ($w_{\mathbf{i}p}^n = N(\mathbf{x}_p^n - \mathbf{x}_\mathbf{i})$) that describe the degree of interaction between particle p and grid node $\mathbf{i}$ and which are defined by Eulerian grid interpolation functions $N(\mathbf{x})$, (2) the grid momentum ($m_\mathbf{i}^n\mathbf{v}_\mathbf{i}^n$) is updated in a variational way from the potential energy in the system and finally, (3) the motion of the grid under the updated momentum is interpolated to the particles. The process of updating the grid momentum in step (2) uses the updated Lagrangian [BLM13, JST16, GW03] convention where the time t^n configuration serves as the reference, rather than the $t = 0$ configuration in a Lagrangian discretization. With this updated Lagrangian convention, the particles $\mathbf{x}_p^n$ are moved by the grid via interpolation $\mathbf{x}_p^{n+1} = \sum_\mathbf{i} \mathbf{x}_\mathbf{i}^{n+1} w_{\mathbf{i}p}^n$, and they change the potential energy via the per-element deformation gradient computed as in standard FEM (see Equation (4.10)). The grid node vertices $\mathbf{x}_\mathbf{i}$, which are allowed to move temporarily as $\mathbf{x}_\mathbf{i}^{n+1} = \mathbf{x}_\mathbf{i} + \Delta t \mathbf{v}_\mathbf{i}^{n+1}$, serve as degrees of freedom. When the spatial discretization is done variationally from the potential energy, this step is almost identically what is done in a Lagrangian FEM discretization of elastoplasticity [SB12]. In this sense, the method can be interpreted as continually remeshing the domain of the material, where the transfer process in step (1) is all that is needed to define the mesh at a given time step (see Figure 4.6). We refer the reader to [JSS15, JST16] for more basic MPM details.

The MPM update only considers the variation of the potential energy with respect to grid degrees of freedom; nothing explicit is done to model self collision. Self collision is modeled as if it were an elastic phenomenon, and by virtue of switching between particle and grid representations. We describe these two aspects of collision resolution as **type (i)** and **type (ii)**.

Type (i) The grid transfers in step (1) ultimately remesh the domain (see Figure 4.6). By transferring to the grid, and using an updated Lagrangian formulation where the grid nodes are updated based on the variation of the potential energy in Equation (4.6), MPM essentially uses a new FEM mesh (blue in Figure 4.6) to calculate the elastic update. This process creates new connections in the updated Lagrangian mesh and once they are made, collision inducing modes are penalized via the potential energy in the system (see Figure 4.6). For example, collision trajectories of the particles will induce compression in elements of the Eulerian grid which would be penalized from the elastic potential in the system.

Type (ii) In particle systems, collisions occur because of discontinuities in the velocity, e.g. consider two particles next to each other with opposing velocities. Transferring to and from the grid smooths the particle velocities, which ultimately prevents collision. Since the motion of the Eulerian grid after the momentum update in step (2) is interpolated to the particles using continuous interpolating functions, particle collisions cannot occur as long as the Eulerian mesh is not tangled by the motion. This can be guaranteed with a CFL restriction since the tangling is a temporal discretization artifact. In fact, an updated Lagrangian MPM simulation with no constitutive model on the particles at all can still prevent material collision, simply by virtue of the **type (ii)** interactions (see Figure 4.7).

These modes of collision resolution are simplistic, but limited by several drawbacks. For volumetric objects, the **type (i)** interactions are unable to regulate the potential energy with a plasticity model derived from Coulomb friction as in [JGT17, GHF18]. The mesh is volumetric and therefore does not have the flexibility of codimension that can be used to model contact through the continuum. There are no directions left for plastic flow of the type designed in [JGT17] that could be used to satisfy the Coulomb friction stress constraints. This can lead to unregulated resistance to shearing and cohesion as the elastic potential will still increase with these modes, even though that is not consistent with Coulomb friction (see Figure 4.9). Furthermore, the updated Lagrangian treatment of the stress-based momentum leads to visual interaction at a distance and persistent wrinkling when the grid resolution is too low [JSS15, FGG17, HN17]. Additionally, when the grid resolution is too high, **type (i)** and **type (ii)** interactions have no effect and the method does not prevent collision (see Figure 4.7). To prevent this, the Lagrangian mesh resolution must be about the same as the Eulerian grid resolution. This is suboptimal when a coarse Lagrangian mesh suffices to resolve deformation.

4.4.1 Hybrid Lagrangian MPM for elastic solids

Our method is designed by abandoning the **type (i)** collision prevention for volumetric meshes and the updated Lagrangian integration of the elastic forces in general. Instead we use a splitting approach where elastic forces are applied in a Lagrangian way, and **type (ii)** interactions are integrated by MPM with no elastic force computation. We achieve this by introducing collision particles $\mathbf{x}_q^n$ which are sampled uniformly at random on the boundary of the volumetric elastic mesh. The mass of the collision particle m_q is found by dividing the mass of the boundary element by the number of collision particles on that element. These particles are not true degrees of freedom and are tied to the mesh during the Lagrangian update. They are then used to generate **type (ii)** collision prevention. We show that their response defines a type of impulse that can be regulated by Coulomb friction and applied

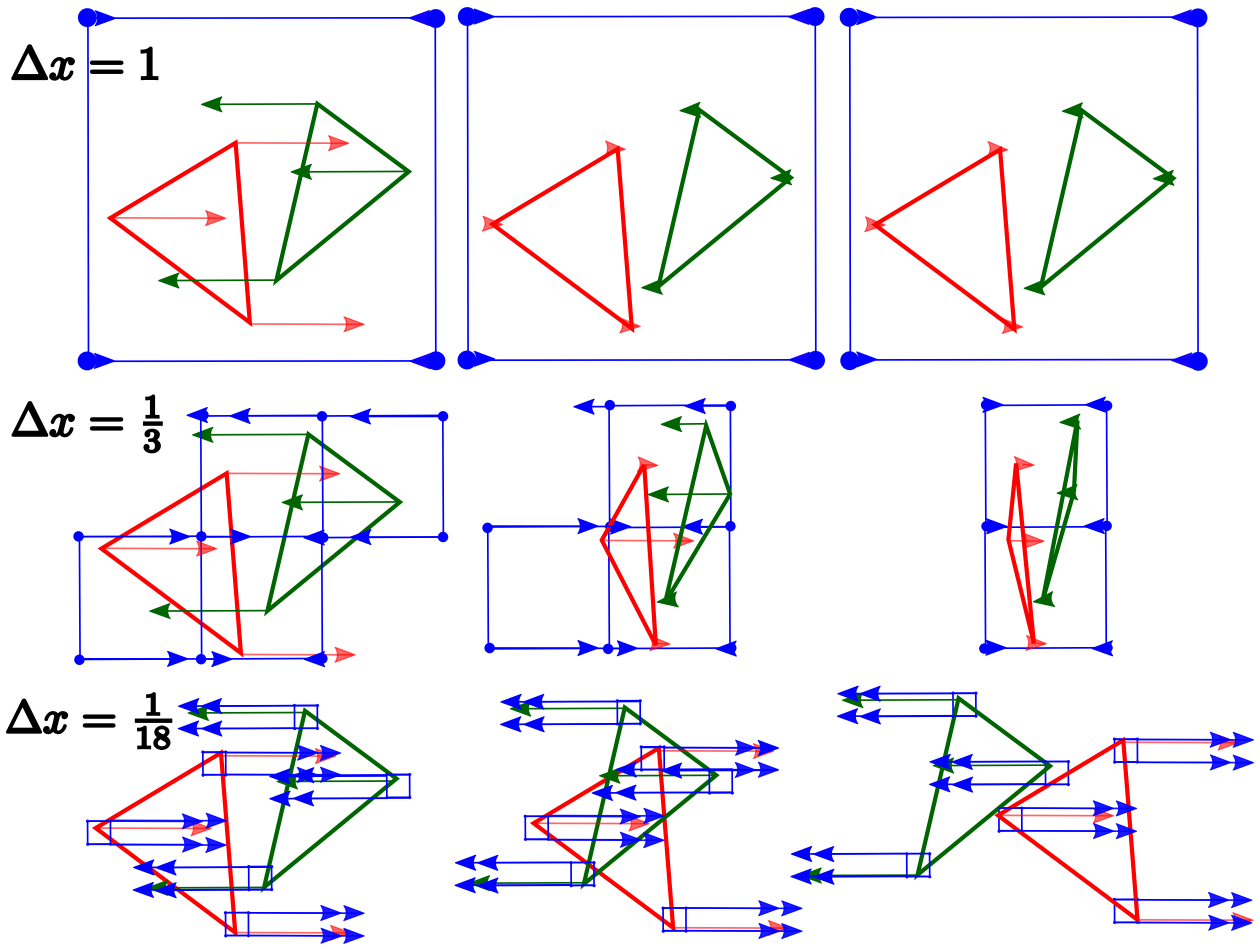

Figure 4.7: **Type (ii) interations with different Δx, columns indicating consecutive time steps**. At appropiate grid resolution (middle row), MPM prevents material collision even without constitutive model. However, when the grid resolution is too low (top row), objects are separated at a distance, and when the grid resolution is too high (bottom row), the MPM grids may miss a collision.

to the mesh at the end of the time step. Furthermore, because the collision particles can be sampled at a density proportional to the grid spacing, we show that they remove the effect of grid resolution on collision resolution (see Figure 4.8).

Our approach uses the same discrete state as in [JSS15]: time t^n, particle positions $\mathbf{x}_p^n$ connected with a tetrahedron mesh, velocities $\mathbf{v}_p^n$, and masses m_p. In addition, we store the collision particles $\mathbf{x}_q^n$ sampled on the boundary of the tetrahedron mesh. We summarize essential steps in the algorithm for updating our discrete state to time t^{n+1} below.

1. **Lagrangian update:** Update particle velocities from potential-energy-based and body forces, and interpolate velocities to collision particles. §4.4.2
2. **Transfer to grid:** Transfer mass and momentum from collision particles to grid. §4.4.3.1
3. **Transfer to collision particles**: Transfer velocities from grid back to collision particles. §4.4.3.2
4. **Apply impulses:** Calculate the impulse applied to each boundary mesh using the velocity change in collision particles and update velocities of particles on the boundary mesh. §4.4.4
5. **Update positions:** Update particle positions and elastic states. §4.4.5.

4.4.2 Lagrangian update

We consider the case of piecewise linear interpolation over a tetrahedron mesh. The deformation gradient varies in a piecewise constant manner with each element, which we denote as $\mathbf{F}_e$. With this convention, the FEM force per particle $\mathbf{f}_p$ can be seen as the negative

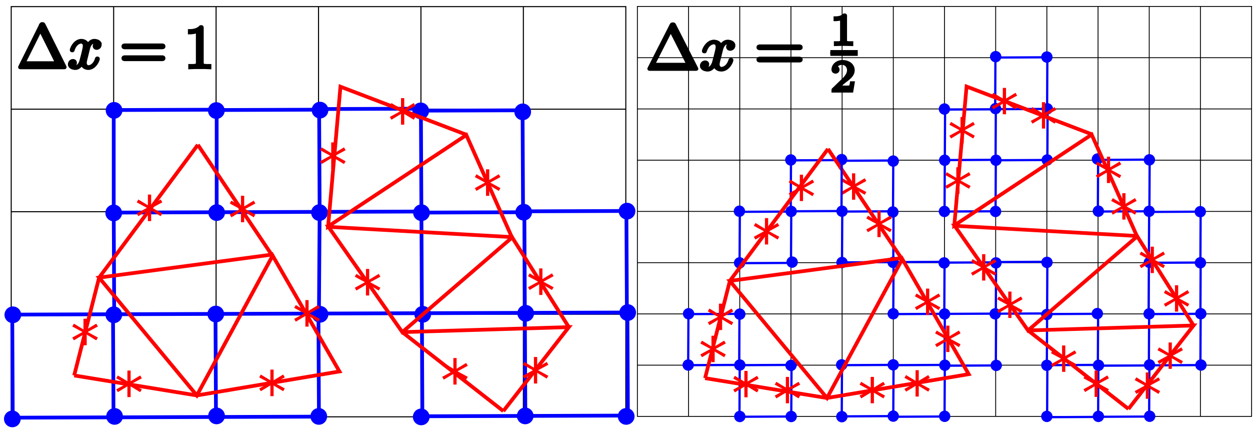

Figure 4.8: **Collision particles**. Sampling density based on Eulerian grid Δx.

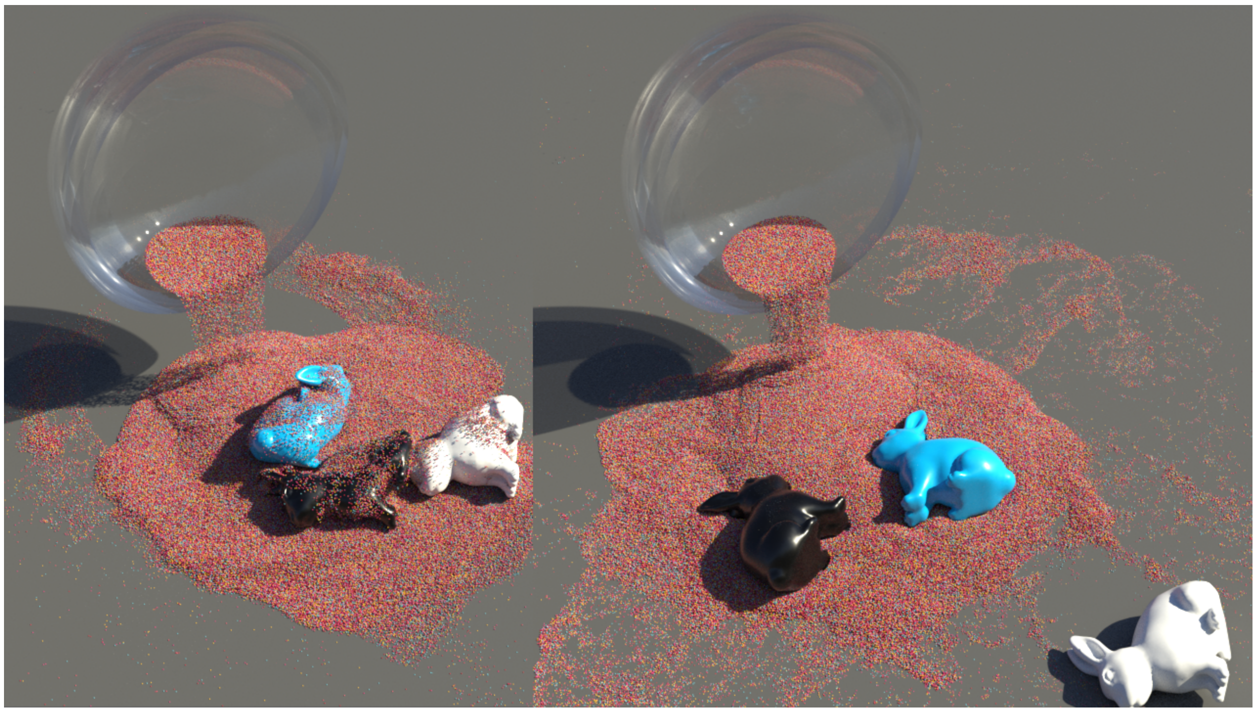

Figure 4.9: **Friction comparison with sand and bunny.** Our method (right) removes the excessive numerical friction common to traditional MPM (left), and regulates friction with the Coulomb friction model. With low friction coefficients, the colored sand freely slides off the bunnies.

gradient of the the total potential energy Ψ with respect to particle positions:

$$\mathbf{F}_e(\mathbf{x}) = \sum_p \mathbf{x}_p \frac{\partial \tilde{N}_p}{\partial \mathbf{X}}(\mathbf{X}_e) \tag{4.10}$$

$$\Psi(\mathbf{x}) = \sum_e \psi(\mathbf{F}_e(\mathbf{x}))V_e^0 \tag{4.11}$$

$$\mathbf{f}_p(\mathbf{x}) = -\sum_e \frac{\partial \psi}{\partial \mathbf{F}}(\mathbf{F}_e(\mathbf{x})) : \frac{\partial \mathbf{F}_e}{\partial \mathbf{x}_p}(\mathbf{x})V_e^0 \tag{4.12}$$

$$= -\sum_e \mathbf{P}(\mathbf{F}_e(\mathbf{x}))\frac{\partial \tilde{N}_p}{\partial \mathbf{X}}V_e^0. \tag{4.13}$$

Here $\mathbf{x} \in \mathbb{R}^{3n_p}$ refers to the vector of all particles $\mathbf{x}_p$, where n_p is the total number of particles, Ψ is the total potential energy which is a sum of tetrahedron element contributions $\psi(\mathbf{F}_e)V_e^0$, where ψ is the potential energy density in Equation (4.3), V_e^0 is the volume of the element in the initial state, $\tilde{N}_p$ is the piecewise linear interpolating function associated with particle $\mathbf{x}_p$, and $\mathbf{X}_e$ is the tetrahedron barycenter in the time $t = 0$ configuration. We refer the reader to Sifakis and Barbic [SB12] for a more detailed derivation.

The FEM update uses the usual Lagrangian view of the governing physics. The internal force is the negative gradient of the potential energy in Equation (4.13). Particle velocities are updated according to forces computed at particle positions $\mathbf{x}_p^{n+\alpha}$, where symplectic Euler integration corresponds to $\alpha = 0$ and backward Euler corresponds to $\alpha = 1$:

$$\mathbf{v}_p^* = \mathbf{v}_p^n + \Delta t \frac{\mathbf{f}_p(\mathbf{x}^{n+\alpha})}{m_p}. \tag{4.14}$$

When damping is required while using symplectic Euler integration, we construct a background Eulerian grid with Δx comparable to the mesh size and transfer the velocity to and then back from the grid using APIC with RPIC damping as described in [JGT17]. We can even perform the transfers multiple times when more damping is desired. For interior particles, $\mathbf{v}_p^{n+1} = \mathbf{v}_p^*$. On the other hand, for particles on the boundary mesh, we interpolate

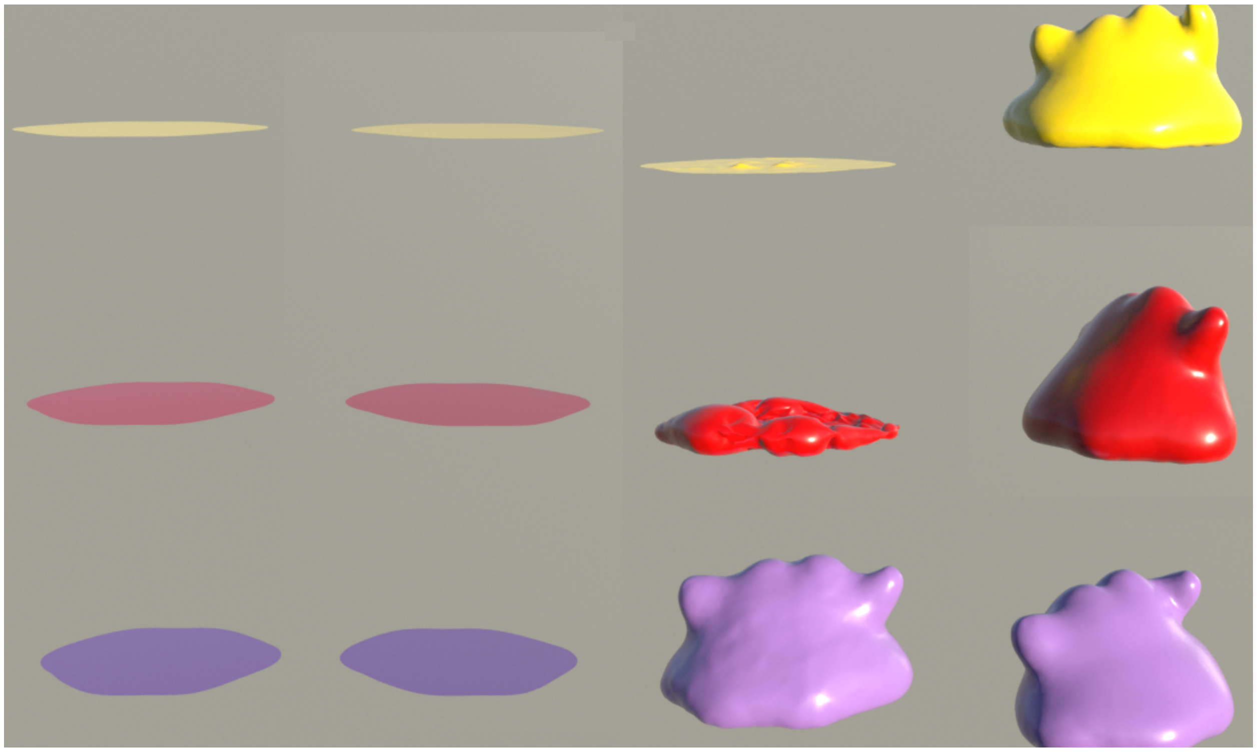

Figure 4.10: **Element inversion**. MPM (left) has difficulties when elements invert, especially with low grid resolution (yellow and red). Our method (right) handles element inversions with ease.

their velocities and positions to collision particles using

$$\mathbf{v}_q^* = \sum_p b_{pq} \mathbf{v}_p^* \tag{4.15}$$

$$\mathbf{x}_q^n = \sum_p b_{pq} \mathbf{x}_p^n \tag{4.16}$$

where b_{pq} is the barycentric weight of the point q relative to p. We also assign to each point q an outward normal vector $\mathbf{n}_q$ inherited from the face of the mesh that q is tied to.

4.4.3 Grid transfers

4.4.3.1 Particle to Grid

To process collision and contact, we transfer mass and momentum from collision particles $\mathbf{x}_q^n$ to grid nodes $\mathbf{x}_\mathbf{i}$ using standard MPM transfers

$$m_\mathbf{i}^n = \sum_q w_{\mathbf{i}q}^n m_q \tag{4.17}$$

$$\mathbf{v}_\mathbf{i}^* = \frac{1}{m_\mathbf{i}^n} \sum_q w_{\mathbf{i}q}^n m_q \mathbf{v}_q^*. \tag{4.18}$$

Here $w_{\mathbf{i}q}^n = N(\mathbf{x}_q^n - \mathbf{x}_\mathbf{i})$ is the weight of interaction between particle $\mathbf{x}_q^n$ and grid node $\mathbf{x}_\mathbf{i}$, as in standard MPM.

4.4.3.2 Grid to Particle

Without any constitutive model on the grid, we proceed directly to the grid to particle step. The grid to particle transfer defines the velocity local to collision particle $\mathbf{x}_q^n$ in terms of $\mathbf{v}_q^\star$ from

$$\mathbf{v}_q^\star = \sum_\mathbf{i} w_{\mathbf{i}q}^n \mathbf{v}_\mathbf{i}^*. \tag{4.19}$$

4.4.4 Apply impulse

Since the velocity $\mathbf{v}_q^\star$ is interpolated from an updated Lagrangian background grid, the boundary of the mesh is safe from self-intersection if it is moved with $\mathbf{v}_q^\star$. However, the change may not be consistent with a Coulomb friction interaction, and the response can even be cohesive. In the case of a cohesive response after collision, we reject the change.

That is, when

$$\mathbf{v}_r = \mathbf{v}_q^\star - \mathbf{v}_q^* \tag{4.20}$$

$$\mathbf{v}_r \cdot \mathbf{n}_q \geq 0 \tag{4.21}$$

the updated Lagrangian mesh detects a separation instead of collision, and the collision particle keeps the velocity from the FEM update $\mathbf{v}_q^*$. On the other hand, if

$$\mathbf{v}_r \cdot \mathbf{n}_q < 0 \tag{4.22}$$

we apply an elastic impulse $I_q \mathbf{n}_q$ to the mesh at position $\mathbf{x}_q^n$ where $I_q = 2m_q \mathbf{v}_r \cdot \mathbf{n}_q$. We also allow for friction using Coulomb's model with the friction parameter μ. When an elastic impulse of magnitude I_q would be applied based on condition (4.22), Coulomb friction admits a change in magnitude of tangential velocity of at most $-\mu \frac{I_q}{m_q}$. So the combined velocity change on collision particle q is then

$$\Delta \mathbf{v}_q = \frac{I_q \mathbf{n}_q}{m_q} + \min\left(\|\mathbf{v}_t\|, -\mu \frac{I_q}{m_q}\right) \frac{\mathbf{v}_t}{\|\mathbf{v}_t\|}, \tag{4.23}$$

where $\mathbf{v}_t = \mathbf{v}_r - \mathbf{v}_r \cdot \mathbf{n}_q \mathbf{n}_q$. We then transfer this change to the particles p as

$$\Delta \mathbf{v}_p = \mathbf{v}_p^{n+1} - \mathbf{v}_p^* = \sum_q \tilde{b}_{pq} \Delta \mathbf{v}_q \tag{4.24}$$

where

$$\tilde{b}_{pq} = \frac{b_{pq} m_q}{\sum_r b_{pr} m_r} \tag{4.25}$$

are the normalized weights defined from the barycentric weights used to transfer from particles to collision particles.

4.4.5 Update positions and elastic state

For boundary particles, we adopt symplectic Euler time integration

$$\mathbf{v}_p^{n+1} = \mathbf{v}_p^n + \Delta \mathbf{v}_p \tag{4.26}$$

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \Delta t \mathbf{v}_p^{n+1} \tag{4.27}$$

For interior particles, the update is in accordance with either symplectic Euler or backward Euler, depending on the choice of α in Equation (4.14):

$$\mathbf{v}_p^{n+1} = \mathbf{v}_p^* \tag{4.28}$$

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \Delta t \mathbf{v}_p^{n+1}. \tag{4.29}$$

4.5 Discretization: hair strands

As discussed in Section §4.3.2, we decompose the motion of the hair into that representing individual strand deformation $\boldsymbol{\phi}^s$ and that of frictional sliding and compression $\boldsymbol{\phi}^d$. As in [JGT17, GHF18], we discretize these two motions in different ways. Since $\boldsymbol{\phi}^s$ only considers single hair strands, it suffices to discretize the energy and forces with traditional FEM. We do this using the approach of [BWR08, BAV10]. However, unlike the approaches in [JGT17, GHF18], we do not make use of an updated Lagrangian discretization of $\boldsymbol{\phi}^s$. To do so severely limits the ability of the hair to resolve collisions without a prohibitively high-resolution Eulerian grid (see Figure 4.11). Rather, we split the updates of $\boldsymbol{\phi}^s$ and $\boldsymbol{\phi}^d$, where the velocities for $\boldsymbol{\phi}^s$ are first updated in a Lagrangian manner and $\boldsymbol{\phi}^d$ with a standard updated Lagrangian MPM discretization. We then adopt the approach of McAdams et al. [MSW09] where the grid-based updates are interpreted as impulsive changes in velocities on the strand that prevent self collision. However, by foregoing the updated Lagrangian discretization of $\boldsymbol{\phi}^s$, we cannot guarantee that self collision is prevented and thus revert to geometric impulses after the correction from $\boldsymbol{\phi}^d$.

The discrete state for each strand at time t^n consists of centerline particle positions $\mathbf{x}_p^n$, with velocities $\mathbf{v}_p^n$, masses m_p, APIC matrix $\mathbf{C}_p^n$, and elastic and plastic deformation gradients associated with $\boldsymbol{\phi}^d$, $\mathbf{F}_p^{E,n}$ and $\mathbf{F}_p^{P,n}$. Furthermore, each edge e connecting particles $\mathbf{x}_e^n$ and $\mathbf{x}_{e+1}^n$ stores orientation angle θ_e as in [BAV10]. We summarize essential steps in the algorithm for updating the discrete state to time t^{n+1} below.

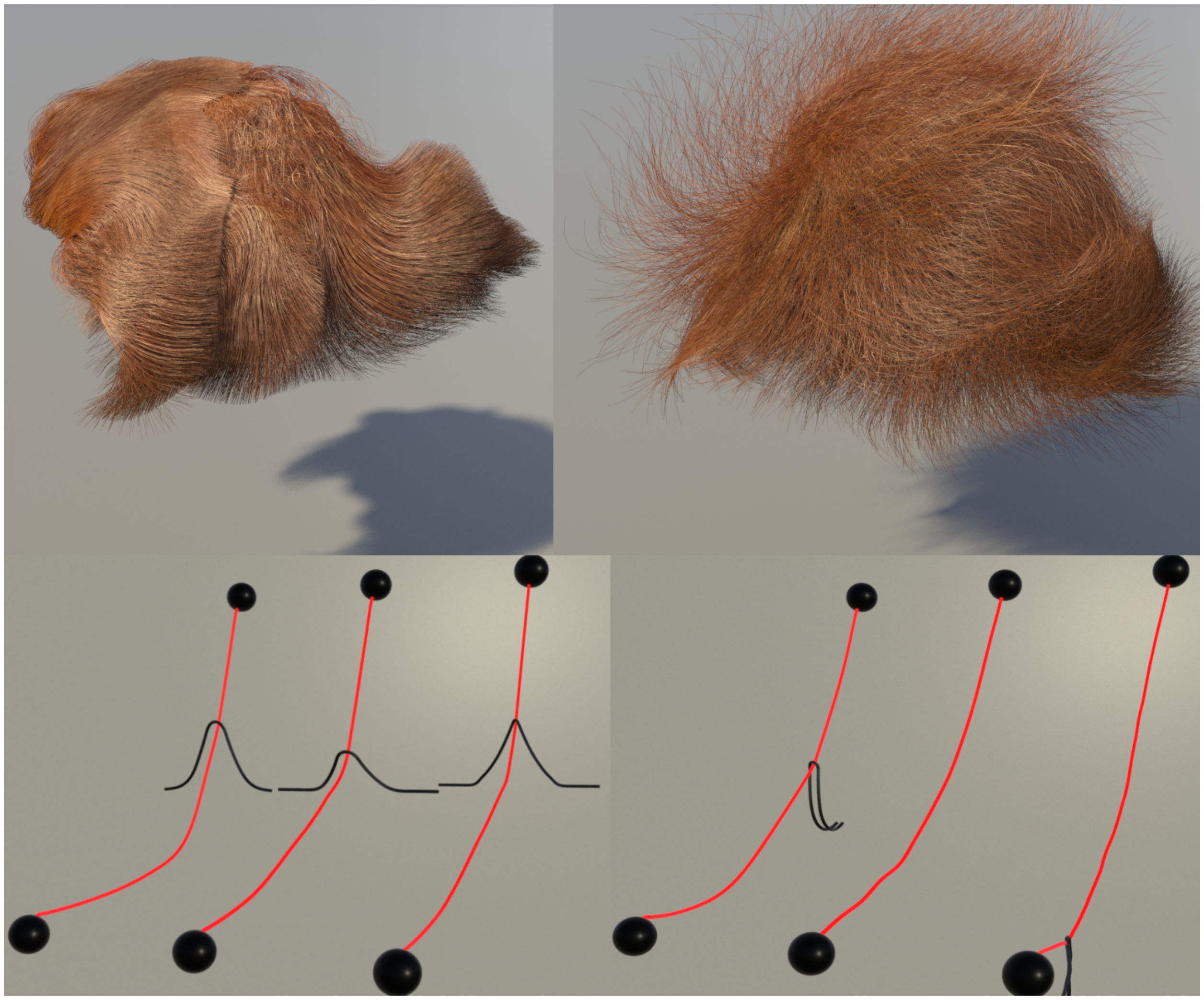

Figure 4.11: **Hair comparison with MPM.** Top row: MPM simulation of hair exhibits excessive friction and cohesion whereas our method captures the rich dynamics of individual strands. Bottom row: We demonstrate the dynamics of two hair strands, colored black and red, at two time steps. MPM (left) results in uncontrolled friction. Hybrid method without geometric collision (middle) misses the collision. Our method (right) captures the sliding behavior between two strands.

1. **Lagrangian update:** Update particle velocities from strand model of [BAV10]. §4.5.1

2. **Transfer to grid:** Transfer mass and momentum from particles to grid using APIC as in [JSS15].

3. **Update grid momentum:** Compute effect of collision potential and friction elastoplasticity. §4.5.2

4. **Apply impulses:** Interpolate the change in grid velocity to particles and then apply geometric collision handling. §4.5.3

5. **Update positions:** Update particle positions as in Equation (4.29).

4.5.1 Lagrangian update

We adopt a time splitting scheme for the velocity update where the velocity is first updated according to the force induced by the energy Ψ^{DER}. Specifically, we have

$$\mathbf{v}_p^* = \mathbf{v}_p^n + \Delta t \frac{\mathbf{f}_p}{m_p} \tag{4.30}$$

where $\mathbf{f}_p$ is calculated as in [BAV10]. This new velocity $\mathbf{v}_p^*$ is then transferred to the MPM background grid $\mathbf{v}_{\mathbf{i}}^*$ as in Section 4.4.3.1.

4.5.2 Grid momentum update

The grid momentum is then updated according to the elastoplasticity model for the $\boldsymbol{\phi}^s$ motion and associated potential energy Ψ^s:

$$\mathbf{v}_{\mathbf{i}}^\star = \mathbf{v}_{\mathbf{i}}^* - \frac{dt}{m_{\mathbf{i}}^n} \sum_p \frac{\partial \psi^S}{\partial \mathbf{F}^E} (\tilde{\mathbf{F}}_p^E(\tilde{\mathbf{x}}^{n+\alpha}))(\mathbf{F}_p^{E,n})^T \nabla w_{\mathbf{i}p}^n V_P^0 + \Delta t \mathbf{g}. \tag{4.31}$$

Here, $\tilde{\mathbf{F}}_p^E(\tilde{\mathbf{x}}^{n+\alpha})$ is the trial elastic strain and $\tilde{\mathbf{x}}^{n+\alpha}$ is the vector of all Eulerian grid node positions, moved according to

$$\mathbf{x}_{\mathbf{i}}^{n+\alpha} = \mathbf{x}_{\mathbf{i}} + \alpha \Delta t \mathbf{v}_{\mathbf{i}}^\star,\ \tilde{\mathbf{F}}_p^E = (\mathbf{I} + \alpha \Delta t \sum_{\mathbf{i}} \mathbf{v}_{\mathbf{i}}^\star \nabla w_{\mathbf{i}p}^n) \mathbf{F}_p^{E,n} \tag{4.32}$$

where $\alpha = 0$ corresponds to symplectic Euler and $\alpha = 1$ corresponds to backward Euler for the grid momentum update. We also update APIC matrix $\mathbf{C}_p$ using grid velocity $\mathbf{v}_{\mathbf{i}}^{\star}$ as in [JSS15, JST16].

4.5.3 Impulses

To interpret the motion in $\mathbf{v}_{\mathbf{i}}^{\star}$ as inducing impulsive change in momentum on the midline, we interpolate the change in the grid velocity to the particles. However, we blend in the updated Lagrangian response weighted with parameter ξ

$$\mathbf{v}_p^{\star} = (1-\xi)\left(\mathbf{v}_p^{*} + \sum_{\mathbf{i}} \left(\mathbf{v}_{\mathbf{i}}^{\star} - \mathbf{v}_{\mathbf{i}}^{*}\right) w_{\mathbf{i}p}^{n}\right) + \xi \sum_{\mathbf{i}} \mathbf{v}_{\mathbf{i}}^{\star} w_{\mathbf{i}p}^{n}. \tag{4.33}$$

This introduces a bit of the **type (i)** and **type (ii)** collision prevention, but without sacrificing the geometric detail of the Lagrangian motion. This is equivalent to the PIC/FLIP blend used in [MSW09]. Typically, we introduce $\xi = 0.95$. However, abandoning the updated Lagrangian update can leave collisional modes unresolved for hair. We apply geometric collision handling similar to [BFA02] to resolve remaining collisional modes.

Collision impulses are applied based on proximity between strand edges. We use acceleration structures for efficient proximity queries as in [BFA02]. However, we use regular grid-based structures inherent in MPM implementations. We divide the domain into calculation pads in space with edge length l. Then we extend the pad in the positive axis direction by proximity threshold δ so that neighboring pads have an overlap of length at least δ and thus any proximity pair will appear in at least one pad. In parallel, each extended pad collects all segments that have at least one endpoint contained in the pad, and then registers any proximity pairs contained in its set of segments. We apply an impulse to any proximity pair on a colliding trajectory as determined by relative velocity component on the direction separating the pair. The inelastic impulses from [BFA02] are then calculated and distributed to particles. Also as proposed in Bridson et al. [BFA02], we divide the total impulse on a particle by the number of impulses it receives from all pads and perform Jacobi iteration.

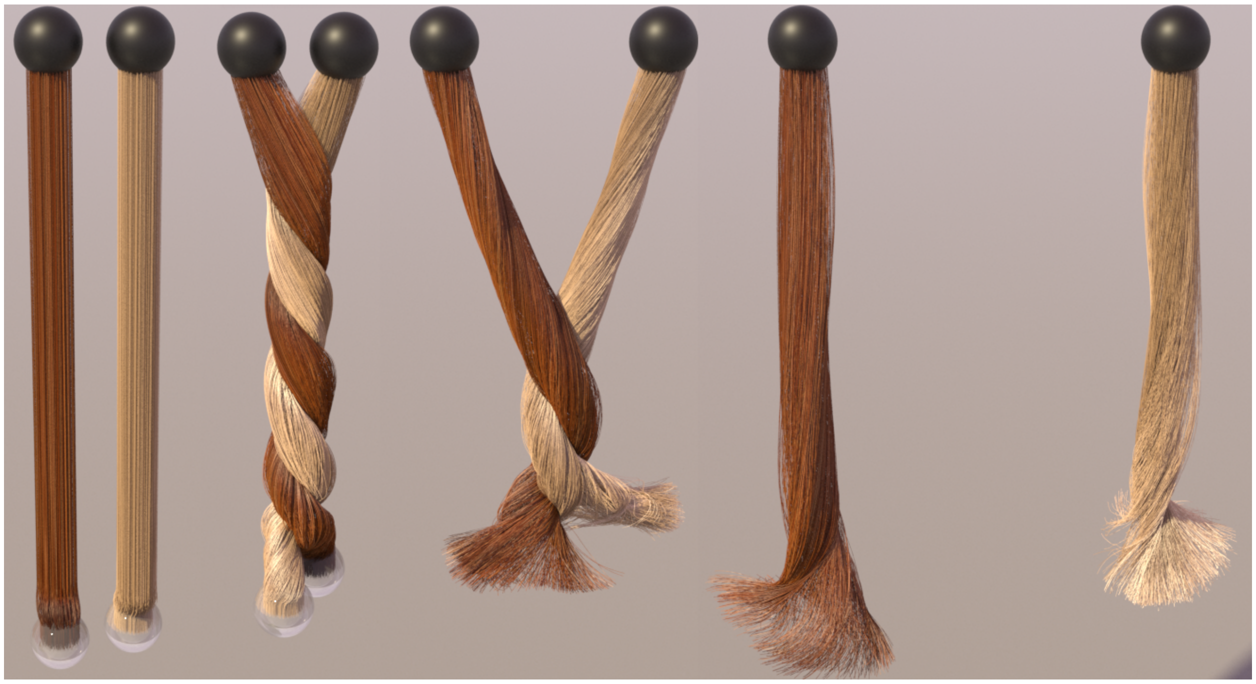

Figure 4.12: **Braiding hair.** Two bundles of hair are interwined into a braid and then separated.

After a fixed number of iterations, we obtain the particle velocity $\mathbf{v}_p^{n+1}$, and then advect particles using Equation (4.29).

4.6 Rigid bodies

Two-way rigid body coupling may be achieved with a treatment similar to volumetric elastic objects. We sample collision particles on the boundary in the same fashion as in Section 4.4.1 and then uniformly distribute the mass of the rigid body to the collision particles. However, we found that unlike for volumetric elastic objects, **type (ii)** interactions on the grid alone are not enough to resolve collisions. Instead we endow the collision particles with the potential described in [JGT17, GHF18] to penalize contact. Specifically, we update the deformation gradient $\mathbf{F}_q$ from time t_n to t_{n+1} in the following way. Let $\mathbf{x}_\alpha$ and $\mathbf{X}_\alpha$, $\alpha \in \{0, 1, 2\}$ be the current and initial positions of the vertices of the triangle that collision particle q is tied to.

Let $\mathbf{D}_{q,\beta} = \mathbf{X}_\beta - \mathbf{X}_0$ be the undeformed mesh element edge vectors (where $\beta = 1, 2$), and $\hat{\mathbf{d}}^E_{q,\beta} = \mathbf{x}^n_\beta - \mathbf{x}^n_0$ be the deformed edge vectors. We choose each $\mathbf{D}_3$ to be unit-length and normal to $\mathbf{D}_1$ and $\mathbf{D}_2$, and evolve each one as in traditional MPM via $\hat{\mathbf{d}}^E_{q,3} = \boldsymbol{\nabla}\mathbf{x}_q \mathbf{d}^E_{q,3}$. Then $\hat{\mathbf{F}}^E_q = \hat{\mathbf{d}}^E_q \mathbf{D}^{-1}_q$. Following [JGT17, GHF18], we let $\hat{\mathbf{F}}^E_q = \mathbf{Q}\hat{\mathbf{R}}$ be the QR decomposition of $\hat{\mathbf{F}}^E_q$ and design a collision energy density $\psi(\hat{\mathbf{R}}) = f(\hat{\mathbf{R}}) + g(\hat{\mathbf{R}})$,

$$f(\hat{\mathbf{R}}) = \begin{cases} \frac{k^c}{3}(1-\hat{r}_{33})^3 & 0 \le \hat{r}_{33} \le 1 \\ 0 & \hat{r}_{33} > 1 \end{cases}, \quad g(\hat{\mathbf{R}}) = \frac{\gamma}{2}(\hat{r}^2_{13} + \hat{r}^2_{23}) \tag{4.34}$$

where $\hat{r}_{ij}$ is the ij-th entry of $\hat{\mathbf{R}}$. We resolve the force which is the negative derivative of this energy on the MPM background grid, and we refer the reader to [JSS15, JST16] for more details. Plasticity is then applied according to [JGT17, GHF18] to give $\mathbf{R}$

$$r_{33} = \begin{cases} \hat{r}_{33} & 0 < \hat{r}_{33} \le 1 \\ 1 & \hat{r}_{33} > 1 \end{cases}, \quad r_{\beta 3} = h(\hat{r}_{13}, \hat{r}_{23}, r_{33})\hat{r}_{\beta 3} \tag{4.35}$$

$$h(\hat{r}_{13}, \hat{r}_{23}, r_{33}) = \min\left(1, \frac{c_F k^c (1 - r_{33})^2}{\gamma\sqrt{\hat{r}^2_{13} + \hat{r}^2_{23}}}\right) \tag{4.36}$$

Finally, we update the deformation gradient with $\mathbf{F}^{n+1}_q = \mathbf{Q}\mathbf{R}$.

Let $\mathbf{v}^*_q = \sum_{\mathbf{i}} w^n_{\mathbf{i}q}\mathbf{v}^*_{\mathbf{i}}$, where $\mathbf{v}^*_{\mathbf{i}}$ is the grid velocity after the MPM force update, and let $\mathbf{v}_r = \mathbf{v}^*_q - \mathbf{v}_q$. If $\mathbf{v}_r \cdot \mathbf{n}_q < 0$, we apply an impulse $\mathbf{I}_q$ to the rigid bodies to update velocity $\mathbf{v}$ and angular velocity $\boldsymbol{\omega}$ via

$$I_q = m_q \mathbf{v}_r \cdot \mathbf{n}_q \tag{4.37}$$

$$\mathbf{v}_t = \mathbf{v}_r - \mathbf{v}_r \cdot \mathbf{n}_q \mathbf{n}_q \tag{4.38}$$

$$\mathbf{I}_q = I_q \mathbf{n}_q + m_q \min\left(\|\mathbf{v}_t\|, -\mu\frac{I_q}{m_q}\right)\frac{\mathbf{v}_t}{\|\mathbf{v}_t\|} \tag{4.39}$$

$$\mathbf{v}^{n+1} = \mathbf{v}^n + \sum_q \frac{\mathbf{I}_q}{m_q} \tag{4.40}$$

$$\boldsymbol{\omega}^{n+1} = \boldsymbol{\omega}^n + \sum_q \mathbf{J}^{-1}(\mathbf{r} \times \mathbf{I}_q) \tag{4.41}$$

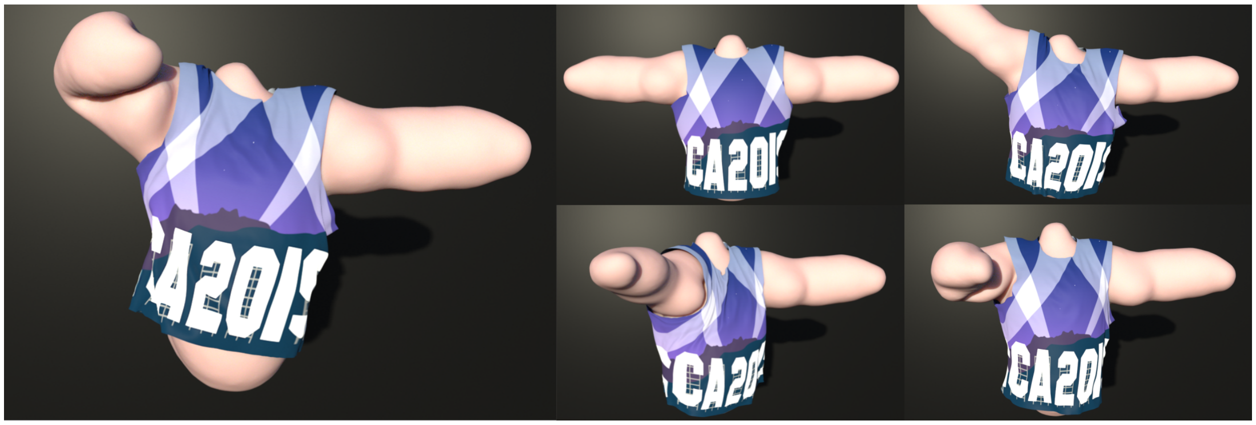

Figure 4.13: **Skin and shirt.** The skin of a mannequin is coupled with clothing simulated with MPM.

where $\mathbf{r}$ is the vector from the rigid body's center of mass to the application point of the impulse, and $\mathbf{J}$ is the inertia tensor.

4.7 Coupling with traditional MPM

Our method easily couples with traditional MPM particles such as snow, sand and clothing. To prevent numerical cohesion between phases common to MPM, we adopt two separate background MPM grids, one for volumetric elastic and rigid objects, and the other for general MPM materials. We denote quantities associated with the two grids with subscripts 1 and 2 respectively. We denote quantities associated with traditional MPM particles with subscript p and quantities associated with quadrature points with subscript q. So we have,

$$m_{1,\mathbf{i}}^n = \sum_q w_{\mathbf{i}q}^n m_q, m_{2,\mathbf{i}}^n = \sum_p w_{\mathbf{i}p}^n m_p \tag{4.42}$$

$$\mathbf{v}_{1,\mathbf{i}}^* = \frac{1}{m_{1,\mathbf{i}}^n} \sum_q w_{\mathbf{i}q}^n m_q \mathbf{v}_q^* \tag{4.43}$$

$$\mathbf{v}_{2,\mathbf{i}}^n = \frac{1}{m_{2,\mathbf{i}}^n} \sum_p w_{\mathbf{i}p}^n m_p \left(\mathbf{v}_p + \mathbf{C}_p(\mathbf{x}_\mathbf{i} - \mathbf{x}_p)\right) \tag{4.44}$$

$$\mathbf{n}_\mathbf{i}^n = \frac{\sum_q w_{\mathbf{i}q}\mathbf{n}_q}{\|\sum_q w_{\mathbf{i}q}\mathbf{n}_q\|} \tag{4.45}$$

Grid velocity $\mathbf{v}_{2,\mathbf{i}}^n$ is updated as in [JSS15, JST16] to get $\mathbf{v}_{2,\mathbf{i}}^*$. Then the collision between phases is handled through an inelastic collision on collocated grid nodes.

$$\mathbf{v}_r = \mathbf{v}_{1,\mathbf{i}}^* - \mathbf{v}_{2,\mathbf{i}}^* \tag{4.46}$$

$$\mathbf{v}_t = \mathbf{v}_r - \mathbf{v}_r \cdot \mathbf{n}_\mathbf{i}^n \mathbf{n}_\mathbf{i}^n \tag{4.47}$$

$$I_\mathbf{i} = \max\left(\frac{m_{2,\mathbf{i}}^n m_{1,\mathbf{i}}^n}{m_{2,\mathbf{i}}^n + m_{1,\mathbf{i}}^n}\mathbf{v}_r \cdot \mathbf{n}_\mathbf{i}^n, 0\right) \tag{4.48}$$

$$\mathbf{v}_{1,\mathbf{i}}^{**} = \mathbf{v}_{1,\mathbf{i}}^* - \frac{I_\mathbf{i}\mathbf{n}_\mathbf{i}}{m_{1,\mathbf{i}}^n} - \min\left(\frac{\mu I_\mathbf{i}}{m_{1,\mathbf{i}}^n}, \|\mathbf{v}_t\|\right)\frac{\mathbf{v}_t}{\|\mathbf{v}_t\|} \tag{4.49}$$

$$\mathbf{v}_{2,\mathbf{i}}^{n+1} = \mathbf{v}_{2,\mathbf{i}}^* + \frac{I_\mathbf{i}\mathbf{n}_\mathbf{i}}{m_{2,\mathbf{i}}^n} + \min\left(\frac{\mu I_\mathbf{i}}{m_{2,\mathbf{i}}^n}, \|\mathbf{v}_t\|\right)\frac{\mathbf{v}_t}{\|\mathbf{v}_t\|} \tag{4.50}$$

Finally, we interpolate the the grid velocity $\mathbf{v}_{2,\mathbf{i}}^{n+1}$ to MPM particles with APIC as in [JSS15, JST16], and Equation (4.19) is replaced with

$$\mathbf{v}_q^\star = \sum_\mathbf{i} w_{\mathbf{i}q}^n \mathbf{v}_{1,\mathbf{i}}^{**}. \tag{4.51}$$

4.8 Results

We demonstrate the efficacy of our method with a number of representative examples that illustrate the dynamics of hair and volumetric objects, and show that our method couples with granular materials, clothing and rigid bodies. We list the runtime performance for

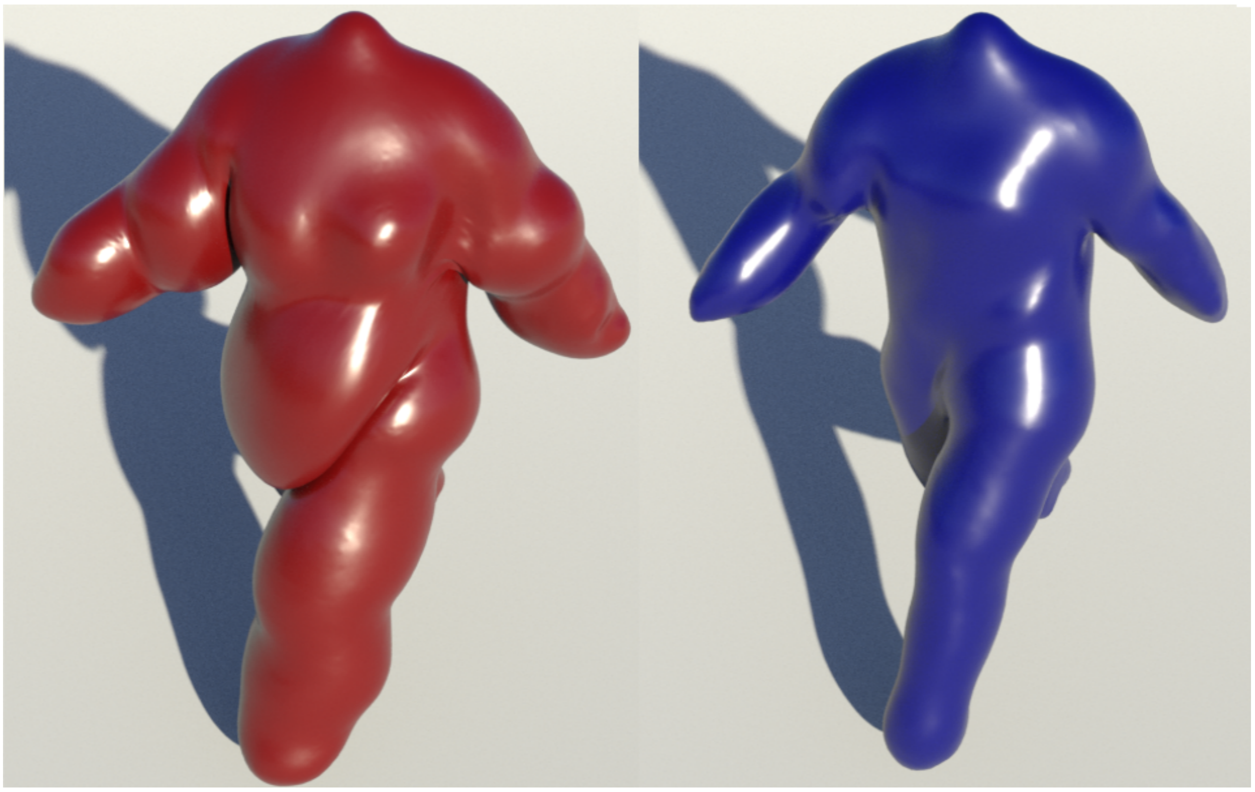

Figure 4.14: **Walking mannequins.** Our method handles the numerous collisions occurring in the scene with walking characters.

our examples in Table 4.1. All simulations were run on an Intel Xeon E5-2690 V2 system with 20 threads and 128GB of RAM. We report the timing in terms of average seconds of computation per frame. We chose Δt in an adaptive manner that is restricted by a CFL condition when the particle velocities are high, i.e., we do not allow particles to move further than the CFL number times Δx in a time step.

4.8.1 Hair

We demonstrate that our method preserves the intricate dynamics of individual hair strands and robustly handles the numerous collisions among them. In Figure 4.11, 32 thousand strands of hair with 60 segments per strand are simulated subject to intense boundary motions. Our algorithm is able to run this challenging example at 122 seconds per frame. In Figure 4.3 and Figure 4.12, we show that our method effortlessly resolves the intense self collisions occurring in braiding examples. In Figure 4.5 and Figure 4.1 (middle), we show a mannequin with a full head of hair in motions common in everyday life, such as walking and dancing. In Figure 4.15, we compare our method with McAdams et al. [MSW09] in a numerical experiment where a bundle of hair strands falls and bounces off another bundle. The experiments are run with a total of 2700 hair strands with 175 segments per strand. Five iterations of impulse application are applied to resolve the collisions missed by advecting the segments with the velocity in Equation (4.33) in our method and the velocity satisfying incompressibility condition in [MSW09]. Notice that our method preserves the volume of the hair bundle and does not suffer from numerical cohesion. We run the test for 100 frames until the hair bundles are apparently separated and track the missed collisions in the process by calculating the collision interactions between pairs of segments using the cubic solve proposed in [BFA02]. The test using McAdams et al. [MSW09] registers more than 543 thousand missed collisions whereas the test using our method registers 120 missed collisions. Our method runs three times faster (see Table 4.1). Note that our method not only avoids the expensive Poisson solve for incompressibility, but it also serves as a better approximate collision response and therefore reduces the runtime and number of missed collisions in the collision impulse step. We plot the total energy in the test run with our method in Figure 4.16.

4.8.2 Volumetric objects

We demonstrate the robustness of our method for resolving collisions between volumetric objects. Our method correctly resolves frictional sliding without artifacts. In Figure 4.14, we show a skin simulation with walking characters in various body shapes. In Figure 4.9, we compare our approach with updated Lagrangian MPM, which exhibits excessive cohesion and numerical friction. We also show that our method removes the requirement of comparable grid and mesh resolution. We use a moderate resolution Lagrangian mesh to resolve the dynamics of the bunnies and a high resolution Eulerian grid to resolve more detailed behaviors of the sand. In contrast, updated Lagrangian MPM would require a high resolution Lagrangian mesh for bunnies in order to resolve collisions between phases. Furthermore, traditional MPM methods often have difficulties recovering from element inversions, as the particle modes needed to uninvert the material are lost in the tranfers between particles and the grid due to the **type (ii)** interactions discussed in Section 4.4. On the other hand, our method handles extreme deformation and even element inversion as demonstrated in Figure 4.10. MPM fails to recover the original shape of the object when the grid resolution is low and **type (ii)** interactions are effective and exhibits high frequency noise when the grid resolution is too high for **type (ii)** interactions to be effective. On the other hand, the elastic object recovers its original shape with any grid resolutions using our method.

4.8.3 Coupling with MPM and rigid bodies

Our method also supports coupling with rigid bodies as well as traditional MPM particles such as snow, sand and clothing. In Figure 4.13, we demonstrate the coupling of soft tissues

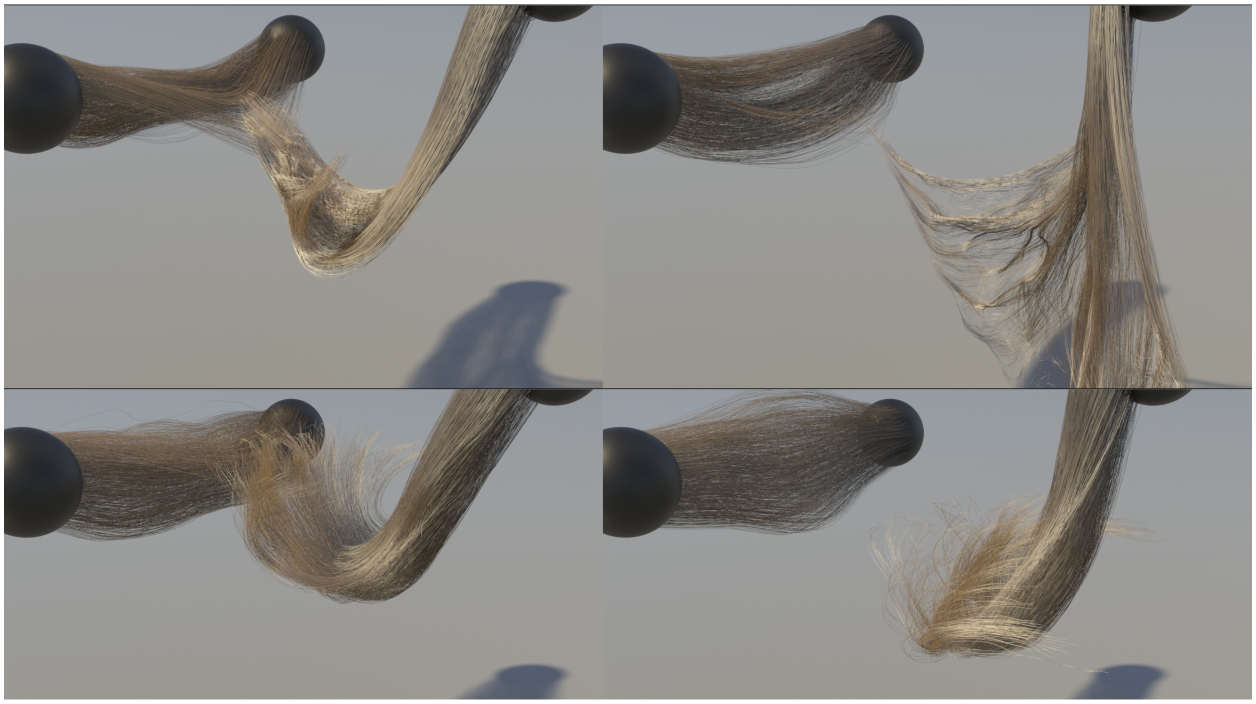

Figure 4.15: **Hair tubes comparison.** Comparison between McAdams et al. [MSW09] (top row) and our method (bottom row) in resolving the collisions between two bundles of hair strands.

with clothing material simulated with MPM as in [JGT17]. In Figure 4.2, we show a hairy ball that is first hit by a snowball and then shakes the snow off. In Figure 4.1 (right), elastic characters and a column of sand are poured on a series of pinwheels simulated as rigid bodies, setting them in motion. In Figure 4.4, colored sand is poured on top of three Jell-O's with various stiffness, generating interesting patterns.

4.9 Discussion and limitations

While our approaches address many shortcomings in existing techniques, there are a number of limitations that persist. First, while abandoning the transversely isotropic elastoplasticity assumption of Jiang et al. [JGT17] does improve the resolution of more complicated strand

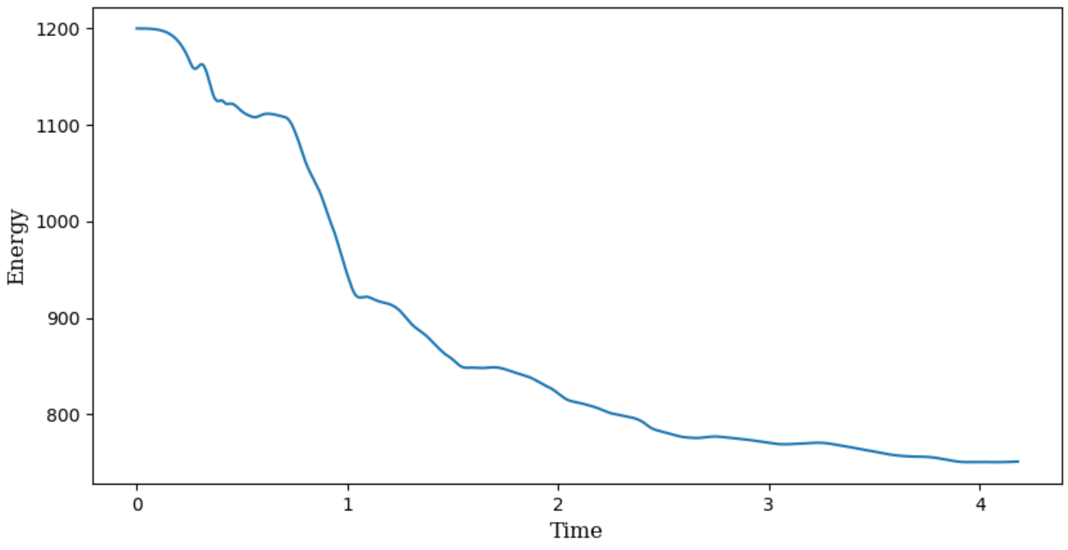

Figure 4.16: **Energy Plot.** We plot the total energy as a function of time for the hair tubes test. The energy is calculated as the sum of elastic and gravitational potential energy and the kinetic energy on the particles.

interactions, as shown in Figure 4.11, it also causes the potential energy associated with collision and shearing to interfere with that of the strand. Interestingly, this does not have an effect under extension. Only under compression of a strand will there be an additional resistance. Furthermore, while our treatment of rigid body dynamics is useful for coupling with elastoplastic materials like sand, soft tissues, etc., our approach is not ideally suited for interactions between rigid bodies. Our approach fails to resolve simple cases like stacking of a few rigid bodies without penetration and/or grid based separation artifacts. Lastly, our collision impulses do not provide any geometric guarantees against self collision, as in e.g. [BFA02]. If large time steps are taken, material will interpenetrate. In general this can be avoided by obeying a CFL condition, as is generally true with MPM.

	Time	Element #	Particle #	Δx	CFL
Mannequin (Fig. 4.14 left)	39	$933K$	$41K/41K$	0.05	0.6
Mannequin (Fig. 4.14 right)	27	$641K$	$31K/31K$	0.05	0.6
Pinwheel (Fig. 4.1 right)	89	$93K$	$930K/57K$	0.5	0.6
Bunnies (MPM) (Fig. 4.9 left)	186	$3.97M$	$2.67M$	0.1	0.6
Bunnies (Hybrid) (Fig. 4.9 right)	66	$201K$	$1.99M/25K$	0.1	0.6
Hair ball (MPM) (Fig. 4.11 top left)	84	$1.92M$	N/A	0.05	0.1
Hair ball (Hybrid) (Fig. 4.11 top right)	122/83	$1.92M$	N/A	0.05	0.1
Hair tubes ([MSW09]) (Fig. 4.15 top)	156/56	$47.5K$	N/A	0.08	0.6
Hair tubes (Hybrid) (Fig. 4.15 bottom)	55/11	$47.5K$	N/A	0.08	0.6
Skin and shirt (Fig. 4.13)	3	$207K$	$120K/40K$	0.006	0.6
Braiding (Fig. 4.12)	87/73	$372K$	N/A	0.15	0.2
Braids (Fig. 4.3)	25/9	$323K$	N/A	0.03	0.2
Hair (Fig. 4.5)	127/46	$1.01M$	N/A	0.05	0.6
Snow on hair (Fig. 4.2)	153/38	$1.92M$	$2.16M$	0.05	0.2
Wall breaking (Fig. 4.1 left)	50	$933K$	$2.29M/41K$	0.05	0.6
Dancer (Fig. 4.1 middle)	117/27	$490K$	N/A	0.04	0.2

Table 4.1: Performance of Hybrid MPM–Lagrangian-FEM Simulations

All simulations were run on an Intel Xeon E5-2690 V2 system with 20 threads and 128GB of RAM. Simulation time is measure in seconds per frame. Time spent on geometric collision per frame is recorded in the second entry of the timing column where applicable. Element # denotes number of segments for hair simulations and number of tetrahedra for volumetric simulations. Particle # denotes the total number of MPM particles, and the number of collision particles are recorded in the second entry where applicable.

REFERENCES

[And17] T. Anderson. *Fracture mechanics: fundamentals and applications.* CRC Press, 2017.

[APK07] B. Adams, M. Pauly, R. Keiser, and L. Guibas. "Adaptively sampled particle fluids." *ACM Trans Graph*, **26**(3), 2007.

[ATW13] R. Ando, N. Thürey, and C. Wojtan. "Highly adaptive liquid simulations on tetrahedral meshes." *ACM Trans Graph*, **32**(4):103:1–103:10, 2013.

[BAC06] F. Bertails, B. Audoly, M. Cani, B. Querleux, F. Leroy, and J. Lévêque. "Super-helices for Predicting the Dynamics of Natural Hair." *ACM Trans Graph*, **25**(3):1180–1187, 2006.

[BAV10] Miklós Bergou, Basile Audoly, Etienne Vouga, Max Wardetzky, and Eitan Grinspun. "Discrete viscous threads." In *ACM Transactions on Graphics (TOG)*, volume 29, p. 116. ACM, 2010.

[BB09] T. Brochu and R. Bridson. "Robust topological operations for dynamic explicit surfaces." *SIAM J Sci Comp*, **31**(4):2472–2493, 2009.

[BCX03] T. Belytschko, H. Chen, J. Xu, and G. Zi. "Dynamic crack propagation based on loss of hyperbolicity and a new discontinuous enrichment." *IntJ Num Meth Eng*, **58**(12):1873–1905, 2003.

[BDP17] M. Buler, P. Diehl, D. Pflger, S. Frey, F. Sadlo, T. Ertl, and M. Schweitzer. "Visualization of fracture progression in peridynamics." *Comp Graph*, **67**(C):45–57, 2017.

[BDW13] O. Busaryev, T. Dey, and H. Wang. "Adaptive fracture simulation of multi-layered thin plates." *ACM Trans Graph*, **32**(4):52:1–52:6, 2013.

[BFA02] R. Bridson, R. Fedkiw, and J. Anderson. "Robust Treatment of Collisions, Contact and Friction for Cloth Animation." *ACM Trans Graph*, **21**(3):594–603, 2002.

[BGB15] H. Bhattacharya, Y. Gao, and A. Bargteil. "A level-set method for skinning animated particle data." *IEEE Trans Vis Comp Graph*, **21**:315–327, 2015.

[BGO06] A. Bargteil, T. Goktekin, J. O'Brien, and J. Strain. "A semi-Lagrangian contouring method for fluid simulation." *ACM Trans Graph*, **25**(1), 2006.

[Bli82] J. Blinn. "A generalization of algebraic surface drawing." *ACM Trans Graph*, **1**(3):235–256, 1982.

[BLM13] T. Belytschko, W. Liu, B. Moran, and K. Elkhodary. *Nonlinear finite elements for continua and structures*. John Wiley and sons, 2013.

[BR86] J. Brackbill and H. Ruppel. "FLIP: A method for adaptively zoned, Particle-In-Cell calculations of fluid flows in two dimensions." *J Comp Phys*, **65**:314–343, 1986.

[BW08] J. Bonet and R. Wood. *Nonlinear continuum mechanics for finite element analysis*. Cambridge University Press, 2008.

[BWH07] A. Bargteil, C. Wojtan, J. Hodgins, and G. Turk. "A finite element method for animating large viscoplastic flow." *ACM Trans Graph*, **26**(3), 2007.

[BWR08] Miklós Bergou, Max Wardetzky, Stephen Robinson, Basile Audoly, and Eitan Grinspun. "Discrete elastic rods." *ACM transactions on graphics (TOG)*, **27**(3):63, 2008.

[Cho14] M. Choi. "Real-time simulation of ductile fracture with oriented particles." *Comp Anim Virt Worlds*, **25**(3-4):457–465, 2014.

[CWX13] F. Chen, C. Wang, B. Xie, and H. Qin. "Flexible and rapid animation of brittle fracture using the smoothed particle hydrodynamics formulation." *Comp Anim Virt Worlds*, **24**(3-4):215–224, 2013.

[CZZ18] W. Chen, F. Zhu, J. Zhao, S. Li, and G. Wang. "Peridynamics-based fracture animation for elastoplastic solids." *Comp Graph Forum*, **37**(1):112–124, 2018.

[DB16] G. Daviet and F. Bertails-Descoubes. "A Semi-implicit Material Point Method for the Continuum Simulation of Granular Materials." *ACM Trans Graph*, **35**(4):102:1–102:13, 2016.

[DBG14] F. Da, C. Batty, and E. Grinspun. "Multimaterial mesh-based surface tracking." *ACM Trans Graph*, **33**(4):112:1–112:11, 2014.

[DC98] M. Desbrun and M. Cani. "Active implicit surface for animation." In *Graph Int*, pp. 143–150, 1998.

[DGP17] F. Dagenais, J. Gagnon, and E. Paquette. "Detail-preserving explicit mesh projection and topology matching for particle-based fluids." *Comp Graph Forum*, **36**(8):444–457, 2017.

[DHW19] Mengyuan Ding, Xuchen Han, Stephanie Wang, Theodore Gast, and Joseph Teran. "A thermomechanical material point method for baking and cooking." *ACM Transactions on Graphics*, **38**:1–14, 11 2019.

[DLC07] N. Daphalapurkar, H. Lu, D. Coker, and R. Komanduri. "Simulation of dynamic crack growth using the generalized interpolation material point (GIMP) method." *Int J Frac*, **143**(1):79–102, 2007.

[FBG18] Y. Fei, C. Batty, E. Grinspun, and C. Zheng. "A multi-scale model for simulating liquid-fabric interactions." *ACM Trans Graph*, **37**(4):51:1–51:16, 2018.

[FGG17] C. Fu, Q. Guo, T. Gast, C. Jiang, and J. Teran. "A Polynomial Particle-in-cell Method." *ACM Trans Graph*, **36**(6):222:1–222:12, November 2017.

[FHH18] Y. Fang, Y. Hu, S. Hu, and C. Jiang. "A temporally adaptive material point method with regional time stepping." In *Computer Graph Forum*, volume 37, pp. 195–204. Wiley Online Library, 2018.

[FLL13] Y. Fan, J. Litven, D. Levin, and D. Pai. "Eulerian-on-lagrangian Simulation." *ACM Trans Graph*, **32**(3):22:1–22:9, 2013.

[FLP14] Y. Fan, J. Litven, and D. Pai. "Active Volumetric Musculoskeletal Systems." *ACM Trans Graph*, **33**(4):152:1–152:9, 2014.

[GBB09] D. Gerszewski, H. Bhattacharya, and A. Bargteil. "A point-based method for animating elastoplastic solids." In *Proc 2009 ACM SIGGRAPH/Eurograph Symp Comp Anim*, pp. 133–138. ACM, 2009.

[GBT07] M. Gissler, M. Becker, and M. Teschner. "Constraint sets for topology-changing finite element models." In *VRIPHYS*, pp. 21–26, 2007.

[GGT18] J. Gaume, T. Gast, J. Teran, A. van Herwijnen, and C. Jiang. "Dynamic anticrack propagation in snow." *Nature Com*, **9**(1):3047, 2018.

[GHF18] Q. Guo, X. Han, C. Fu, T. Gast, R. Tamstorf, and J. Teran. "A material point method for thin shells with frictional contact." *ACM Trans Graph*, **37**(4):147, 2018.

[GN06] J. Guo and J. Nairn. "Three-Dimensional Dynamic Fracture Analysis Using the Material Point Method." *Comp Mod Eng Sci*, **16**, 2006.

[GNL14] A. Golas, R. Narain, and M. Lin. "Continuum modeling of crowd turbulence." *Phys Rev E*, **90**:042816, 2014.

[GPH18] M. Gao, A. Pradhana, X. Han, Q. Guo, G. Kot, E. Sifakis, and C. Jiang. "Animating fluid sediment mixture in particle-laden flows." *ACM Trans Graph*, **37**(4):149:1–149:11, 2018.

[GTJ17] M. Gao, A. Tampubolon, C. Jiang, and E. Sifakis. "An adaptive generalized interpolation material point method for simulating elastoplastic materials." *ACM Trans Graph*, **36**(6):223:1–223:12, 2017.

[GW03] J. E. Guilkey and J. A. Weiss. "Implicit time integration for the material point method: Quantitative and algorithmic comparisons with the finite element method." *Int J Numer Meth Eng*, **57**(9):1323–1338, 2003.

[GWK18] M. Gao, X. Wang, Kui K.Wu, A. Pradhana, E. Sifakis, C. Yuksel, and C. Jiang. "GPU optimization of material point methods." In *SIGGRAPH Asia 2018 Technical Papers*, SIGGRAPH Asia '18, pp. 254:1–254:12, New York, NY, USA, 2018. ACM.

[Har64] F. Harlow. "The particle-in-cell method for numerical solution of problems in fluid dynamics." *Meth Comp Phys*, **3**:319–343, 1964.

[HFG18] Y. Hu, Y. Fang, Z. Ge, Z. Qu, Y. Zhu, A. Pradhana, and C. Jiang. "A moving least squares material point method with displacement discontinuity and two-way rigid body coupling." *ACM Trans Graph*, **37**(4):150:1–150:14, 2018.

[HGG19] Xuchen Han, Theodore Gast, Qi Guo, Stephanie Wang, Chenfanfu Jiang, and Joseph Teran. "A Hybrid Material Point Method for Frictional Contact with Diverse Materials." *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, **2**:1–24, 07 2019.

[HJS13] Jan Hegemann, Chenfanfu Jiang, Craig Schroeder, and Joseph M. Teran. "A Level Set Method for Ductile Fracture." In *Proc ACM SIGGRAPH/Eurograp Symp Comp Anim*, pp. 193–201, 2013.

[HN17] C. Hammerquist and J. Nairn. "A new method for material point method particle updates that reduces noise and enhances stability." *Comp Meth App Mech Eng*, **318**:724 – 738, 2017.

[HZG18] Y. Hu, Q. Zhou, X. Gao, A. Jacobson, D. Zorin, and D. Panozzo. "Tetrahedral Meshing in the Wild." *ACM Trans. Graph.*, **37**(4):60:1–60:14, July 2018.

[HZM11] P. Huang, X. Zhang, S. Ma, and X. Huang. "Contact algorithms for the material point method in impact and penetration simulation." *Int J Num Meth Eng*, **85**(4):498–517, 2011.

[ITF04] G. Irving, J. Teran, and R. Fedkiw. "Invertible Finite Elements for Robust Simulation of Large Deformation." In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim*, pp. 131–140, 2004.

[JGT17] C. Jiang, T. Gast, and J. Teran. "Anisotropic elastoplasticity for cloth, knit and hair frictional contact." *ACM Trans Graph*, **36**(4):152, 2017.

[JML16] B. Jones, A. Martin, J. Levine, T. Shinar, and A. Bargteil. "Ductile fracture for clustered shape matching." In *Proc ACM SIGGRAPH Symp Int 3D Graph Games*, pp. 65–70. ACM, 2016.

[JSS15] C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin. "The Affine Particle-In-Cell Method." *ACM Trans Graph*, **34**(4):51:1–51:10, 2015.

[JST16] Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. "The Material Point Method for Simulating Continuum Materials." In *ACM SIGGRAPH 2016 Course*, pp. 24:1–24:52, 2016.

[KBT17] D. Koschier, J. Bender, and N. Thuerey. "Robust eXtended Finite Elements for complex cutting of deformables." *ACM Trans Graph*, **36**(4):55:1–55:13, 2017.

[KGP16] G. Klár, T. Gast, A. Pradhana, C. Fu, C. Schroeder, C. Jiang, and J. Teran. "Drucker-prager Elastoplasticity for Sand Animation." *ACM Trans Graph*, **35**(4):103:1–103:12, 2016.

[KMB08] P. Kaufmann, S. Martin, M. Botsch, and M. Gross. "Flexible simulation of deformable models using discontinuous Galerkin FEM." In *Proc 2008 ACM SIGGRAPH/Eurograph Symp Comp Anim*, pp. 105–115. Eurographics Association, 2008.

[LHL11] N. Liu, X. He, S. Li, and G. Wang. "Meshless simulation of brittle fracture." *Comp Anim Virt Worlds*, **22**(2-3):115–124, 2011.

[LLJ11] D. Levin, J. Litven, G. Jones, S. Sueda, and D. Pai. "Eulerian Solid Simulation with Contact." *ACM Trans Graph*, **30**(4):36:1–36:10, 2011.

[LSN13] D. Li, S. Sueda, D. Neog, and D. Pai. "Thin Skin Elastodynamics." *ACM Trans Graph*, **32**(4):49:1–49:10, 2013.

[M09] Matthias Müller. "Fast and robust tracking of fluid surfaces." In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim*, pp. 237–245. ACM, 2009.

[MBF04] N. Molino, Z. Bao, and R. Fedkiw. "A virtual node algorithm for changing mesh topology during simulation." *ACM Trans Graph*, **23**(3):385–392, 2004.

[MBT15] O. Mercier, C. Beauchemin, N. Thuerey, T. Kim, and D. Nowrouzezahrai. "Surface turbulence for particle-based liquid simulations." *ACM Trans Graph*, **34(6)**:10, Nov 2015.

[MCG03] M. Müller, D. Charypar, and M. Gross. "Particle-based fluid simulation for interactive applications." In *Proc 2003 ACM SIGGRAPH/Eurograph Symp Comp Anim*, pp. 154–159. Eurographics Association, 2003.

[MCK13] M. Müller, N. Chentanez, and T. Kim. "Real-time dynamic fracture with volumetric approximate convex decompositions." *ACM Trans Graph*, **32**(4):115:1–115:10, 2013.

[MCK15] M. Müller, N. Chentanez, T. Kim, and M. Macklin. "Air Meshes for Robust Collision Handling." *ACM Trans. Graph.*, **34**(4):133:1–133:9, 2015.

[MCZ07] K. Museth, M. Clive, and B. Zafar. "Blobtacular: surfacing particle system in "Pirates of the Caribbean 3"." In *ACM SIGGRAPH 2007 Sketches*, SIGGRAPH '07. ACM, 2007.

[MG04] M. Müller and M. Gross. "Interactive virtual materials." In *Proc Graph Int*, pp. 239–246. Canadian Human-Computer Communications Society, 2004.

[MHH07] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. "Position based dynamics." *J Vis Comm Im Rep*, **18**(2):109–118, 2007.

[MKN04] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. "Point based animation of elastic, plastic and melting objects." In *Proc ACM SIGGRAPH/Eurograp Symp Comp Anim*, pp. 141–151, 2004.

[MSW09] A. McAdams, A. Selle, K. Ward, E. Sifakis, and J. Teran. "Detail Preserving Continuum Simulation of Straight Hair." *ACM Trans Graph*, **28**(3):62:1–62:6, 2009.

[Mus14] K. Museth. *A flexible image processing approach to the surfacing of particle-based fluid animation*, pp. 81–84. 2014.

[Nai03] John A. Nairn. "Material point method calculations with explicit cracks." 2003.

[NGC09] R. Narain, A. Golas, S. Curtis, and M. Lin. "Aggregate Dynamics for Dense Crowd Simulation." *ACM Trans Graph*, **28**(5):122:1–122:8, 2009.

[OBH02] J. O'Brien, A. Bargteil, and J. Hodgins. "Graphical modeling and animation of ductile fracture." In *Proc ACM SIGGRAPH 2002*, pp. 291–294, 2002.

[OH99] J. O'Brien and J. Hodgins. "Graphical modeling and animation of brittle fracture." In *Proc 26th Conf Comp Graph Int Tech*, SIGGRAPH '99, pp. 137–146. ACM Press/Addison-Wesley Publishing Co., 1999.

[OKN09] M. Ohta, Y. Kanamori, and T. Nishita. "Deformation and fracturing using adaptive shape matching with stiffness adjustment." *Comp Anim Virt Worlds*, **20**(2–3):365–373, 2009.

[PKA05] M. Pauly, R. Keiser, B. Adams, P. Dutré, M. Gross, and L. Guibas. "Meshless animation of fracturing solids." *ACM Trans Graph*, **24**(3):957–964, 2005.

[PKK03] M. Pauly, R. Keiser, L. Kobbelt, and M. Gross. "Shape modeling with point-sampled geometry." *ACM Trans Graph*, **22**(3):641–650, 2003.

[PNJ14] T. Pfaff, R. Narain, J. de Joya, and J. O'Brien. "Adaptive tearing and cracking of thin sheets." *ACM Trans Graph*, **33**(4):110:1–110:9, 2014.

[PO09] E. Parker and J. O'Brien. "Real-time deformation and fracture in a game environment." In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim*, pp. 165–175. ACM, 2009.

[Rag02] S. Raghavachary. "Fracture generation on polygonal meshes using Voronoi polygons." In *ACM SIGGRAPH 2002 Conf Abstracts App*, SIGGRAPH '02, pp. 187–187. ACM, 2002.

[RGJ15] D. Ram, T. Gast, C. Jiang, C. Schroeder, A. Stomakhin, J. Teran, and P. Kavehpour. "A material point method for viscoelastic fluids, foams and sponges." In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim*, pp. 157–163, 2015.

[SB12] E. Sifakis and J. Barbic. "FEM simulation of 3D deformable solids: a practitioner's guide to theory, discretization and model reduction." In *ACM SIGGRAPH 2012 Courses*, SIGGRAPH '12, pp. 20:1–20:50, New York, NY, USA, 2012. ACM.

[SCS94] D. Sulsky, Z. Chen, and H. Schreyer. "A particle method for history-dependent materials." *Comp Meth App Mech Eng*, **118**(1):179–196, 1994.

[SHS12] A. Stomakhin, R. Howes, C. Schroeder, and J. Teran. "Energetically consistent invertible elasticity." In *Proc Symp Comp Anim*, pp. 25–32, 2012.

[SMT08] E. Sifakis, S. Marino, and J. Teran. "Globally Coupled Collision Handling Using Volume Preserving Impulses." In *Proc 2008 ACM SIGGRAPH/Eurographics Symp Comp Anim*, pp. 147–153, 2008.

[SO14] S. Schvartzman and M. Otaduy. "Fracture animation based on high-dimensional voronoi diagrams." In *Proc ACM SIGGRAPH Symp Int 3D Graph Games*, pp. 15–22. ACM, 2014.

[SOG09] D. Steinemann, M. Otaduy, and M. Gross. "Splitting meshless deforming objects with explicit surface tracking." *Graph Models*, **71**(6):209–220, 2009.

[SS07] C. Shen and A. Shah. "Extracting and parametrizing temporally coherent surfaces from particles." In *SIGGRAPH Sketches*, p. 66, 2007.

[SSC13] A. Stomakhin, C. Schroeder, L. Chai, J. Teran, and A. Selle. "A Material Point Method for snow simulation." *ACM Trans Graph*, **32**(4):102:1–102:10, 2013.

[SSF09] J. Su, C. Schroeder, and R. Fedkiw. "Energy stability and fracture for frame rate rigid body simulations." In *Proc 2009 ACM SIGGRAPH/Eurograph Symp Comp Anim*, pp. 155–164. ACM, 2009.

[SSI07] E. Sifakis, T. Shinar, G. Irving, and R. Fedkiw. "Hybrid simulation of deformable solids." In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim*, pp. 81–90. Eurographics Association, 2007.

[SSJ14] A. Stomakhin, C. Schroeder, C. Jiang, L. Chai, J. Teran, and A. Selle. "Augmented MPM for phase-change and varied materials." *ACM Trans Graph*, **33**(4):138:1–138:11, 2014.

[SSP07] B. Solenthaler, J. Schläfli, and R. Pajarola. "A unified particle model for Fluid-solid interactions." *Comp Anim Virt Worlds*, **18**(1):69–82, 2007.

[Tao18] Terrence Tao. "254A, Notes 0: Physical derivation of the incompressible Euler and Navier-Stokes equations.", Sep 2018.

[TF88] D. Terzopoulos and K. Fleischer. "Modeling inelastic deformation: viscolelasticity, plasticity, fracture." *SIGGRAPH Comp Graph*, **22**(4):269–278, 1988.

[TGK17] A. P. Tampubolon, T. Gast, G. Klár, C. Fu, J. Teran, C. Jiang, and K. Museth. "Multi-species simulation of porous sand and water mixtures." *ACM Trans Graph*, **36**(4), 2017.

[TLK16] Y. Teng, D. Levin, and T. Kim. "Eulerian Solid-fluid Coupling." *ACM Trans Graph*, **35**(6):200:1–200:8, 2016.

[WAM17] J. Wretborn, R. Armiento, and K. Museth. "Animation of crack propagation by means of an extended multi-body solver for the material point method." *Comp Graph*, **69**(C):131–139, 2017.

[WBG07] M. Wicke, M. Botsch, and M. Gross. "A finite element method on convex polyhedra." *Comp Graph Forum*, **26**:355–364, 2007.

[WDG19a] S. Wang, M. Ding, T. Gast, L. Zhu, S. Gagniere, C. Jiang, and J. Teran. *Supplementary Technical Document*, 2019.

[WDG19b] S. Wang, M. Ding, T. Gast, L. Zhu, S. Gagniere, C. Jiang, and J. Teran. "Simulation and Visualization of Ductile Fracture with the Material Point Method." volume 2, p. 18. ACM, 2019.

[Wil08] B. Williams. *Fluid surface reconstruction from particles.* PhD thesis, University of British Columbia, 2008.

[WRK10] M. Wicke, D. Ritchie, B. Klingner, S. Burke, J. Shewchuk, and J. O'Brien. "Dynamic local remeshing for elastoplastic simulation." *ACM Trans Graph*, **29**(4):49:1–11, 2010.

[WT08] C. Wojtan and G. Turk. "Fast viscoelastic behavior with thin features." *ACM Trans Graph*, **27**(3):1–8, 2008.

[WTG04] M. Wicke, M. Teschner, and M. Gross. "CSG tree rendering for point-sampled objects." In *12th Pac Graph*, pp. 160–168, 2004.

[WTG09] C. Wojtan, N. Thürey, M. Gross, and G. Turk. "Deforming meshes that split and merge." *ACM Trans Graph*, **28**(3):76:1–76:10, 2009.

[WTG10] C. Wojtan, N. Thürey, M. Gross, and G. Turk. "Physics-inspired topology changes for thin fluid features." *ACM Trans Graph*, **29**(4):50:1–50:8, 2010.

[WY16] K. Wu and C. Yuksel. "Real-time Hair Mesh Simulation." In *ACM SIGGRAPH Symp Int 3D Graph Games*. ACM, 2016.

[YSB15] Y. Yue, B. Smith, C. Batty, C. Zheng, and E. Grinspun. "Continuum foam: a material point method for shear-dependent flows." *ACM Trans Graph*, **34**(5):160:1–160:20, 2015.

[YSC18] Y. Yue, B. Smith, P. Chen, M. Chantharayukhonthorn, K. Kamrin, and E. Grinspun. "Hybrid grains: adaptive coupling of discrete and continuum simulations of granular media." *ACM Trans Graph*, **37**(6):283:1–283:19, 2018.

[YT13] J. Yu and G. Turk. "Reconstructing surfaces of particle-based fluids using anisotropic kernels." *ACM Trans. Graph.*, **32**(1):5:1–5:12, 2013.

[YWT12] J. Yu, C. Wojtan, G. Turk, and C. Yap. "Explicit mesh surfaces for particle based fluids." *Comp Graph Forum*, **31**(2pt4):815–824, 2012.

[ZB05] Y. Zhu and R. Bridson. "Animating sand as a fluid." *ACM Trans Graph*, **24**(3):965–972, 2005.

[ZJ10] C. Zheng and D. James. "Rigid-body fracture sound with precomputed soundbanks." *ACM Trans Graph*, **29**(4):69:1–69:13, 2010.

[ZZL17] F. Zhu, J. Zhao, S. Li, Y. Tang, and G. Wang. "Dynamically enriched MPM for invertible elasticity." In *Comp Graph Forum*, volume 36, pp. 381–392. Wiley Online Library, 2017.

[ZZS06] N. Zhang, X. Zhou, D. Sha, X. Yuan, K. Tamma, and B. Chen. "Integrating mesh and meshfree methods for physics-based fracture and debris cloud simulation." In M. Botsch, B. Chen, M. Pauly, and M. Zwicker, editors, *Symp Point-Based Graph*. Eurograph Assoc, 2006.