UNIVERSITY OF CALIFORNIA
Los Angeles

Accurate and Efficient Multi-Material Simulations for Physics-Integrated Digital Twins

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

by

Xuan Li

2025

ABSTRACT OF THE DISSERTATION

Accurate and Efficient Multi-Material Simulations
for Physics-Integrated Digital Twins

by

Xuan Li
Doctor of Philosophy in Mathematics
University of California, Los Angeles, 2025
Professor Chenfanfu Jiang, Chair

This thesis explores research advancements in building accurate physics-integrated digital twins. First, as the core computational engine, accurate and efficient simulations are essential for achieving realistic dynamics. To this end, we develop improved inelasticity simulation frameworks for both the Material Point Method (MPM) and the Finite Element Method (FEM). Additionally, we introduce FEM-MPM coupled simulation frameworks that leverage the strengths of both methods to enable multi-material simulations. Furthermore, we propose efficiency enhancements for cloth and inelasticity simulations. Second, digital twins must be grounded in real-world data. Leveraging recent advancements in neural rendering and differentiable rendering, we propose methods for estimating physical parameters from multiview videos, directly simulating reconstructed environments, immersively interacting with reconstructed worlds, and faithfully reconstructing simulation-ready garments from single-view images. Finally, we demonstrate the critical role of digital twins in mechanical design. Using our simulation-driven approach, we successfully fabricate an aerial vehicle with dual operational modes.

The dissertation of Xuan Li is approved.

Ying Nian Wu

Demetri Terzopoulos

Stanley Osher

Chenfanfu Jiang, Committee Chair

University of California, Los Angeles

2025

To Pingying, my parents, and friends.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor and thesis committee chair, Prof. Chenfanfu Jiang. Throughout this long journey, his brilliance and insightful guidance have shaped me into an independent researcher. I am also grateful to my thesis committee members, Prof. Stanley Osher, Prof. Demetri Terzopoulos and Prof. Ying Nian Wu for their valuable time, support and guidance.

I sincerely appreciate my collaborators: Yue Li, Dr. Minchen Li, Dr. Yixing Zhu, Prof. Bo Zhu, Dr. Jessica Weakly, Prof. Cynthia Sung, Prof. Yin Yang, Prof. Craig Schroeder, Dr. Haozhe Su, Prof. Mridul Aanjaneya, Dr. Yu Fang, Dr. Lei Lan, Dr. Yi-Ling Qiao, Dr. Peter Yichen Chen, Dr. Krishna Murthy Jatavallabhula, Prof. Ming Lin, Dr. Chuang Gan, Yutao Feng, Tianyi Xie, Zeshun Zong, Dr. Yuxing Qiu, Dr. Ying Jiang, Chang Yu, Yunuo Chen, Dr. Donglai Xiang, Dr. Qianli Ma, Dr. Tsung-Yi Lin, Dr. Yongxin Chen. Their collaboration, insights, and support have greatly enriched my research journey.

I am deeply grateful to my parents for their unconditional love, sacrifices, and unwavering support, which have given me the strength and determination to pursue my dreams.

A special thank you to my partner, Pingying Chen, who has witnessed every moment and milestone of my academic journey in the U.S. I would not have achieved what I have today without her unwavering understanding, encouragement, and immense support. I would also like to thank our adorable calico cat, Luna, whose companionship has provided me with endless emotional support throughout this journey.

VITA

2024-2025	Research Intern, NVIDIA Research.
2022–2025	Teaching Assistant, Department of Mathematics, UCLA.
2021–2024	Graduate Student Researcher, Department of Mathematics, UCLA.
2022	Research Intern, MIT-IBM Watson AI Lab.
2020-2021	Research Assistant, Department of CIS, University of Pennsylvania.
2017-2020	M.S. in Computer Science, Stony Brook University.
2019	Research Intern, Adobe Research.
2013-2017	B.S. in Mathematics, Tsinghua University.

PUBLICATIONS

* indicates co-first authors

X. Li*, C. Yu*, W. Du*, Y. Jiang*, T. Xie, Y. Chen, Y. Yang and C. Jiang. 2025. Dress-1-to-3: Single Image to Simulation-Ready 3D Outfit with Diffusion Prior and Differentiable Physics. *arXiv preprint arXiv:2502.03449*.

X. Tan*, Y. Jiang*, **X. Li***, Z. Zong, T. Xie, Y. Yang, and C. Jiang. 2025. PhysMotion: Physics-Grounded Dynamics From a Single Image. *arXiv preprint arXiv:2411.17189*.

B. Li*, **X. Li***, Y. Jiang*, T. Xie, F. Gao, H. Wang, Y. Yang, and C. Jiang. 2025. GarmentDreamer: 3DGS Guided Garment Synthesis with Diverse Geometry and Texture Details. *3DV*.

Y. Chen*, T. Xie*, Z. Zong*, **X. Li**, F. Gao, Y. Yang, Y. N. Wu, and C. Jiang. 2024. Atlas3D: Physically Constrained Self-Supporting Text-to-3D for Simulation and Fabrication. *NeurIPS*.

C. Yu*, **X. Li***, L. Lan, Y. Yang and C. Jiang. 2024. XPBI: Position-Based Dynamics with Smoothing Kernels Handles Continuum Inelasticity. *SIGGRAPH Asia*.

L. Lan, Z. Lu, J. Long, C. Yuan, **X. Li**, X. He, H. Wang, C. Jiang, and Y. Yang. 2024. Efficient GPU Cloth Simulation with Non-distance Barriers and Subspace Reuse. *ACM Transactions On Graphics (SIGGRAPH Asia)*.

Y. Jiang*, C. Yu*, T. Xie*, **X. Li***, Y. Feng, H. Wang, M. Li, H. Lau, F. Gao, Y. Yang and C. Jiang. 2024. VR-GS: A physical dynamics-aware interactive Gaussian splatting system in virtual reality. *SIGGRAPH*.

X. Li, M. Li, X. Han, H. Wang, Y. Yang and C. Jiang. 2024. A Dynamic Duo of Finite Elements and Material Points. *SIGGRAPH*.

T. Xie*, Z. Zong*, Y. Qiu*, **X. Li***, Y. Feng, Y. Yang and C. Jiang. 2024. PhysGaussian: Physics-integrated 3D Gaussians for generative dynamics. *CVPR*.

Y. Feng, Y. Shang, **X. Li**, T. Shao, C. Jiang, and Y. Yang. 2024. Pie-NeRF: Physics-Based Interactive Elastodynamics with NeRF. *CVPR*.

J. Weakly*, **X. Li***, T. Agarwal, M. Li, S. Folk, C. Jiang, and C. Sung. 2024. Bistable Aerial Transformer: A Quadrotor Fixed-Wing Hybrid That Morphs Dynamically Via Passive Soft Mechanism. *Journal of Mechanisms and Robotics*.

X. Li, Y.L. Qiao, P.Y. Chen, K.M. Jatavallabhula, M. Lin, C. Jiang and C. Gan. 2023. PAC-NeRF: Physics Augmented Continuum Neural Radiance Fields for Geometry-Agnostic System Identification. *ICLR*.

X. Li, Y. Fang, L. Lan, H. Wang, Y. Yang, M. Li and C. Jiang. 2023. Subspace-preconditioned GPU projective dynamics with contact for cloth simulation. *SIGGRAPH Asia*.

Z. Zong, **X. Li**, M. Li, M. M. Chiaramonte, W. Matusik, E. Grinspun, K. Carlberg, C. Jiang, and P. Y. Chen. 2023, December. Neural stress fields for reduced-order elastoplasticity and fracture. *SIGGRAPH Asia*.

Y. Fang*, M. Li*, Y. Cao, **X. Li**, J. Wolper, Y. Yang, and C. Jiang. 2023. Augmented Incremental Potential Contact for Sticky Interactions. *TVCG*.

H. Su*, **X. Li***, T. Xue, C. Jiang, and M. Aanjaneya. 2023. A Generalized Constitutive Model for Versatile MPM Simulation and Inverse Learning with Differentiable Physics. *SCA*.

X. Li, Y. Cao, M. Li, Y. Yang, C. Schroeder and C. Jiang. 2022. PlasticityNet: Learning to simulate metal, sand, and snow for optimization time integration. *NeurIPS*.

X. Li, M. Li and C. Jiang. 2022. Energetically consistent inelasticity for optimization time integration. *ACM Transactions on Graphics (SIGGRAPH)*.

X. Li*, Y. Fang*, M. Li and C. Jiang. 2022. BFEMP: Interpenetration-free MPM–FEM coupling with barrier contact. *Computer Methods in Applied Mechanics and Engineering*.

X. Li*, J. Weakly*, M. Li, C. Sung and C. Jiang. 2021. Soft hybrid aerial vehicle via bistable mechanism. *ICRA*.

Y. Li*, **X. Li***, M. Li, Y. Zhu, B. Zhu and C. Jiang. 2021. Lagrangian–Eulerian multidensity topology optimization with the material point method. *International Journal for Numerical Methods in Engineering*.

CHAPTER 1

Introduction

A digital twin is a virtual replica of a physical system that can evolve dynamically according to physical laws and interactions. One of the core foundations of building accurate digital twins is ensuring their physical accuracy. This requires advancements in physics simulation techniques to capture increasingly fine-grained characteristics of real-world dynamic phenomena. With the development of precise physics-based digital twins, industries unlock diverse applications: in VFX, they enhance realism in visual effects; in environmental science, they simulate weather patterns and geological activities to predict natural disasters; in engineering and product design, they accelerate hardware iteration and optimize performance before production; and in robotics, they provide a safe, efficient environment for AI training, refining skills in virtual simulations before real-world deployment.

On the other hand, visual representation is also crucial for creating truly accurate digital twins. This requires them to be grounded in real-world data. Recent advances in neural scene reconstruction have significantly reduced the manual effort needed to generate virtual assets that visually replicate physical objects. While these methods excel at capturing geometric structures and textures, they often require extensive refinement to be effectively simulated in virtual environments. Bridging this gap requires integrating physics into reconstructions, making digital twins not only look realistic but also simulation-ready.

This dissertation explores innovative research directions that address both physical accuracy and real-world data grounding, aiming to develop highly precise, physics-integrated digital twins.

1.1 Thesis Contributions and Overview

Accurate Inelasticity Modeling.

Inelasticity is widely observed in everyday objects. However, a loss of accuracy occurs in almost all existing work on inelasticity modeling. In Section 3.1, we present Energetically Consistent Inelasticity (ECI), a novel inelastic material modeling approach that augments hyperelastic energy density functions to enable robust, fully implicit elastoplasticity and viscoelasticity simulations. We derive a new fully implicit inelastic force formulation and discover analytical variational energy forms for von-Mises plasticity and viscoelasticity. The force model can be extended to J2 plasticity with some modifications to the integrator. These models can be incorporated into both the Finite Element Method (FEM) and the Material Point Method (MPM). However, the analytical variational formulations are currently derived for only a limited number of inelasticity models. To address this limitation, in Section 3.2, we propose PlasticityNet, a neural network-based elastoplastic modeling framework for arbitrary inelasticity models. PlasticityNet represents elastoplastic forces as the positional gradients of learned potential energies, eliminating the need for tedious analytical derivations or expensive nonlinear root-finding methods without significantly sacrificing accuracy.

FEM-MPM Coupled Simulation.

The Finite Element Method (FEM) has achieved notable success in animating elastic objects, including solids, shells, and rods. However, it encounters challenges when dealing with severe deformations, such as topology changes induced by plasticity, where the Material Point Method (MPM) excels. This contrast between FEM and MPM highlights the need for their coupling in complex multi-material simulations to leverage their respective strengths. In Section 4.1, we propose BFEMP, which establishes the foundation for coupled simulations by treating MPM particles as physical entities and employing contact forces induced by sphere-triangle collisions to achieve two-way interactions between the two domains. This framework is evaluated under FEM-MPM elasticity coupling with an optimized time integrator. However, while FEM benefits from implicit integration, MPM is more efficient under explicit schemes, where plasticity handling is inherently "plug-and-play." To enable an implicit-explicit coupled

system, in Section 4.2, we propose Dynamic Duo, a novel framework designed to seamlessly integrate implicit FEM with explicit MPM. This is achieved through an asynchronous time-splitting scheme, where FEM elasticity and FEM-MPM contact are coupled implicitly, while the contact force is applied explicitly to MPM particles. Dynamic Duo achieves up to a 200× difference in respective time step sizes.

Towards Efficient Simulation.

The efficiency of simulations has long been a significant challenge. While increasing computing power has gradually reduced running times over the years, advancements in algorithmic efficiency can lead to even more dramatic improvements in performance. In Section 5.1, we propose SPPD, an efficient cloth simulation method based on the Projective Dynamics (PD) framework on modern GPUs. Our approach combines subspace integration with parallelizable iterative relaxation techniques to effectively reduce both high-frequency and low-frequency residuals, leading to significantly improved convergence. In Section 5.2, we introduce XPBI, a method that combines continuum inelasticity constitutive modeling with eXtended Position-Based Dynamics (XPBD), a widely used real-time simulation technique. Unlike ECI, XPBI requires only the force gradient to reach equilibrium, which significantly improves the efficiency of inelasticity simulations while maintaining similar accuracy.

Real-to-Sim Tasks.

To fully reconstruct physics-integrated digital twins, both accurate physical parameters and realistic asset appearances are essential. Recent advancements in neural scene reconstruction have bridged the gap between the real and virtual worlds using only RGB observations. However, these reconstructions generally lack physics-based interactivity. One crucial missing piece is the estimation of physical parameters. In Section 6.1, we present PAC-NeRF, a framework that estimates physical material parameters from multi-view videos, leveraging the differentiability provided by neural radiance fields (NeRFs). Our continuum reformulation of NeRFs enables the use of differentiable simulation to evolve radiance fields, thereby creating a differentiable pipeline from material parameters to 2D visual renderings. However, this

evolution of implicit volumes results in some loss of appearance fidelity. With the emergence of 3D Gaussian Splatting (3DGS), we discovered that its explicit representation is extremely well-suited for simulations, inspiring PhysGaussian, introduced in Section 6.2. By applying physical deformations induced by the Material Point Method (MPM) directly to 3DGS representations, we achieve both realistic dynamics and high-fidelity rendering in an unified framework. A key method for immersive interaction with digital twins is virtual reality (VR). In Section 6.3, we deploy VR-GS, a PhysGaussian-like system in VR, by innovatively incorporating real-time mesh-based XPBD simulations with 3DGS for real-time, physics-integrated immersive interactions. Despite significant breakthroughs in neural reconstruction, a major challenge remains: dense multi-view observations are required, limiting applicability to casually captured shots. In Section 6.4, we tackle this challenge within the domain of garment reconstruction from a single image. Following the paradigm of PAC-NeRF, our approach reconstructs both physics parameters and geometry via streaming differentiable simulation and differentiable rendering. Instead of relying on multi-view observations, we introduce a novel approach that leverages multi-view diffusion models, trained on internet-scale datasets, to generate novel views from a single input image.

Sim-to-Real Applications.

One important application of digital twins is predicting future outcomes in virtual environments. This capability allows us to develop optimal strategies before deploying them in real-world scenarios. In mechanical engineering, simulation plays a crucial role. In Section 7.1, we introduce a novel approach that incorporates differentiable MPM into nonlinear topology optimization, which aims to determine the optimal shape with maximum resistance to a given external load. This framework enables the development of a novel hybrid aerial vehicle (HAV), presented in Section 7.2. Through topology optimization, we successfully fabricate a bistable arm structure that allows the HAV to transition between fixed-wing mode and quadrotor mode without requiring additional actuators to maintain each respective mode.

CHAPTER 2

Background

2.1 Continuum Motion and Governing Laws

In continuum mechanics, the motion of a deformable body is described by a deformation map $\boldsymbol{x} = \Phi(\boldsymbol{X}, t)$, which tracks the position $\boldsymbol{x}$ of a material point initially located at $\boldsymbol{X}$ in the reference configuration Ω^0 as the body evolves over time. The deformation gradient $\boldsymbol{F} = \frac{\partial \Phi}{\partial \boldsymbol{X}}(\boldsymbol{X}, t)$ measures the local distortions of infinitesimal material elements. It provides a complete description of how small regions of the continuum body stretch, rotate, and shear during motion. When such local distortions occur, elastic forces arise to resist deformation. These internal resistive forces are characterized by the stress tensor, which encapsulates the response of the material to deformation. The motion of a continuum body is governed by two fundamental conservation laws.

$$R(\boldsymbol{X}, t)J(\boldsymbol{X}, t) = R(\boldsymbol{X}, 0), \qquad \text{(Mass Conservation)}, \tag{2.1a}$$

$$R(\boldsymbol{X}, 0)\frac{\partial \boldsymbol{V}}{\partial t}(\boldsymbol{X}, t) = \nabla^{\boldsymbol{X}} \cdot \boldsymbol{P} + R(\boldsymbol{X}, 0)\boldsymbol{g}, \qquad \text{(Momentum Conservation)}, \tag{2.1b}$$

where $R(\boldsymbol{X}, t)$ is the mass density tracked on material particles in the deformed state, $J(\boldsymbol{X}, t) := \det \boldsymbol{F}(\boldsymbol{X}, t)$ is the Jacobian determinant of the deformation gradient, representing the local volume change due to deformation, $\boldsymbol{V}(\boldsymbol{X}, t) := \frac{\partial \Phi(\boldsymbol{X}, t)}{\partial t}$ is the tracked velocity field defined on material points, $\boldsymbol{P}(\boldsymbol{F})$ is the first-Piola Kirchoff stress,representing the internal forces exerted per unit reference area, and $\boldsymbol{g}$ is the gravity acceleration vector. Elastic materials are defined using constitutive models, specifically through different expressions of stress $\boldsymbol{P}(\boldsymbol{F})$.

2.2 Discretization

2.2.1 Temporal Discretization and Time Integration

To numerically solve Equation (2.1b), we discretize time by introducing evenly spaced discrete time steps:

$$t^0, t^1, t^2, ..., t^n, t^{n+1}, ...$$

where the time step size is denoted as Δt. Since the momentum equation involves a time derivative of velocity, we approximate the velocity derivative using finite difference:

$$\frac{\partial \boldsymbol{V}(\boldsymbol{X}, t)}{\partial t} \approx \frac{1}{\Delta t}(\boldsymbol{V}(\boldsymbol{X}, t^{n+1}) - \boldsymbol{V}(\boldsymbol{X}, t^n)) \tag{2.2}$$

The process of advancing the solution from $\boldsymbol{V}(\boldsymbol{X}, t^n)$ to $\boldsymbol{V}(\boldsymbol{X}, t^{n+1})$ is known as time integration. Depending on how we approximate the time-dependent stress term, there are two primary approaches: explicit methods evaluate stress at time t^n, while implicit methods evaluate stress at time t^{n+1}. Explicit methods are computationally efficient but require small time steps, which may lead to long overall running times. In contrast, implicit methods typically require solving a nonlinear equation system but offer long-term stability and allow for larger time steps.

2.2.2 Spatial Discretization

2.2.2.1 Finite Element Method

Finite Element Method (FEM) discretize the material domain Ω^0 using meshes. We solve the momentum equation using the integral form:

$$\frac{1}{\Delta t}\int_{\Omega^0} R(\boldsymbol{X}, 0)(\boldsymbol{V}^{n+1} - \boldsymbol{V}^n) w_\alpha d\boldsymbol{X} = -\int_{\Omega^0} \boldsymbol{P}\nabla^{\boldsymbol{X}} w_\alpha d\boldsymbol{X} + \int_{\Omega_0} R(\boldsymbol{X}, 0)\boldsymbol{g} w_\alpha d\boldsymbol{X} \tag{2.3}$$

where w_α is an arbitrary test function defined on discrete meshes. When using linear test functions defined on mesh nodes, the above integral equation can be converted to the following

discrete momentum equation:

$$\boldsymbol{M}_i(\boldsymbol{V}_i^{n+1} - \boldsymbol{V}_i^n - \boldsymbol{g}\Delta t) = -\Delta t \sum_e \boldsymbol{P}(\boldsymbol{F}_e^*)\nabla w_{ie} V_e^0. \tag{2.4}$$

Here $* \in \{n, n+1\}$ depends on the time integrator, $\boldsymbol{M}_i$ represents the lumped mass at vertex i, $\boldsymbol{V}_i$ denotes the discretized velocity field at vertex i, ∇w_{ie} is the gradient of the test function at vertex i evaluated on mesh element e, and V_e^0 is the initial volume of element e. For linear-basis FEM, the deformation gradient $\boldsymbol{F}_e$ is the local affine transformation on mesh element e from the initial configuration to the current deformed configuration. Note that mass conservation is trivial if we assume the vertex mass remains constant over time.

We usually use implicit integrators in FEM due to their stability, where the stress term is evaluated at t^{n+1}. And the deformation gradient is evaluated via the following approximations

$$\boldsymbol{V}_i^{n+1} = \frac{1}{\Delta t}(\boldsymbol{X}_i^{n+1} - \boldsymbol{X}_i^{n+1}) \Longrightarrow \boldsymbol{X}_i^{n+1} = \boldsymbol{X}_i^n + \Delta t \boldsymbol{V}_i^{n+1} \tag{2.5}$$

$$\boldsymbol{F}_e^{n+1} = \sum_i \boldsymbol{X}_i^{n+1} \nabla w_{ie}^T \tag{2.6}$$

Consequently, we need to solve a large nonlinear equation system with the number of degrees of freedom (DoFs) corresponding to the vertices. When the velocity updated, we update vertex positions according to Equation (2.5).

Optimization Time Integrators Among implicit integrators, the optimization-based time integrator is particularly favored due to its robustness. For hyperelastic materials, such as the fixed corotated and Neo-Hookean models, the first Piola-Kirchhoff stress can be expressed as the gradient of a corresponding energy density function:

$$\boldsymbol{P}(\boldsymbol{F}) = \frac{\partial \Psi}{\partial \boldsymbol{F}}(\boldsymbol{F}). \tag{2.7}$$

The discrete implicit momentum equation, Equation (2.4), can be reformulated as the following optimization problem:

$$\boldsymbol{V}^{n+1} = \arg\min_{\boldsymbol{V}} \frac{1}{2}\|\boldsymbol{V} - \boldsymbol{V}^n - \boldsymbol{g}\Delta t\|_{\boldsymbol{M}} + \sum_e \boldsymbol{\Psi}(\boldsymbol{F}_e)V_e^0 \tag{2.8}$$

This optimization problem can be robustly solved using Newton's method with line search to ensure convergence.

Projective Dynamics Projective Dynamics (PD) (Bouaziz et al., 2014) introduces an efficient approach to solving the above optimization for specific elastic energy functions. For example, consider the fixed-corotated elastic energy:

$$\Psi(\boldsymbol{F}) = \mu\|\boldsymbol{F} - \boldsymbol{R}\|^2 + \frac{1}{2}\lambda(J-1)^2, \tag{2.9}$$

where R is the closest rotation matrix to F. The optimization is solved using a sequence of global-local iterations. At each local step i, we project the current deformation gradient $\boldsymbol{F}_e$ for each element onto its closest rotation matrix $\boldsymbol{R}_e^i$ and its closest matrix with determinant 1, $\boldsymbol{K}_e^i$. At each global step i, we solve the following quadratic optimization problem:

$$\boldsymbol{V}^{n+1,i} = \arg\min_{\boldsymbol{V}} \frac{1}{2}\|\boldsymbol{V} - \boldsymbol{V}^n - \boldsymbol{g}\Delta t\|_{\boldsymbol{M}} + \mu\sum_e \|\boldsymbol{F}_e - \boldsymbol{R}^i\|^2 V_e^0 + \frac{1}{2}\sum_e \|\boldsymbol{F}_e - \boldsymbol{K}_e^i\|^2. \tag{2.10}$$

This alternating procedure is iterated until convergence. The efficiency of PD arises from two key aspects: the local step can be executed in parallel for each element, significantly improving computational efficiency; the global step solves a quadratic system with a constant coefficient matrix across the entire simulation, allowing for efficient prefactorization.

2.2.2.2 Material Point Method

The Material Point Method (MPM) discretizes the continuum body using a set of Lagrangian particles. Instead of solving the momentum equation with Ω^0 as the reference configuration,

it pushes forward Equation (2.3) to the current configuration $\Omega^n = \Phi(\Omega^0)$:

$$\frac{1}{\Delta t}\int_{\Omega^n} \rho(\boldsymbol{x}, t^n)(\hat{\boldsymbol{v}}^{n+1} - \boldsymbol{v}^n) w_\alpha^n d\boldsymbol{x} = -\int_{\Omega^n} \frac{1}{J^n}\boldsymbol{P}\boldsymbol{F}^{nT}\nabla^{\boldsymbol{x}} w_\alpha^n d\boldsymbol{x}. \tag{2.11}$$

Here, w_α^n is an arbitrary B-spline test function defined on a regular Eulerian grid, where the grid spacing Δx is user-defined. Note that $\hat{\boldsymbol{v}}^{n+1}(\boldsymbol{x}, t)$, $x \in \Omega^n$ is not the velocity field defined on Ω^{n+1}; instead, it represents the tracked velocity on material points in Ω^n. Integrals are evaluated by treating Lagrangian particles as quadrature points. The corresponding discretized momentum equation is:

$$m_i^n(\hat{\boldsymbol{v}}_i^{n+1} - \boldsymbol{v}_i^n - \boldsymbol{g}\Delta t) = -\Delta t \sum_p \boldsymbol{P}(\boldsymbol{F}_p^*)\boldsymbol{F}_p^{nT}\nabla w_{ip}^n V_p^0, \tag{2.12}$$

where m_i is the lumped mass at grid node, $\boldsymbol{v}, \hat{\boldsymbol{v}}$ are the velocity fields discretized on the grid, and ∇w_{ip} is the gradient of the test function defined on grid i, evaluated at particle p. Note that $V_p^0 = V_p^n/J^n$ with V_p being the current volume of particle p. The grid is implicitly constructed at each time step, and $m_i^n, \boldsymbol{v}^n$ are transferred from Lagrangian particles to the grid.

Both explicit and implicit MPM methods are commonly used. In explicit MPM, the stress tensor is evaluated using $\boldsymbol{F}_p^n$. In implicit MPM, we assume a velocity-position relationship similar to Equation (2.5):

$$\hat{\boldsymbol{x}}^{n+1} = \boldsymbol{x}^n + \Delta t \hat{\boldsymbol{v}}^{n+1}, \tag{2.13}$$

which leads to the following trial deformation gradient for stress tensor evaluation:

$$\boldsymbol{F}^{\mathrm{tr},n} = \boldsymbol{F}^n + \Delta t \nabla^{\boldsymbol{x}} \hat{\boldsymbol{v}}^{n+1} = \boldsymbol{F}^n + \Delta t \sum_i \hat{\boldsymbol{v}} \nabla w_{ip}^T. \tag{2.14}$$

After solving for the new grid velocity field $\hat{\boldsymbol{v}}^{n+1}$, the velocity field is transferred from the grid back to the particles, and the particles are advected accordingly.

We summarize the MPM time integration as follows:

1. **Transfer Particles to Grid.** Transfer mass and momentum from particles to grids as

$$\begin{aligned} m_i^n &= \sum_p w_{ip}^n m_p, \\ m_i^n \boldsymbol{v}_i^n &= \sum_p w_{ip}^n m_p \left(\boldsymbol{v}_p^n + \boldsymbol{C}_p^n \left(\boldsymbol{x}_i - \boldsymbol{x}_p^n \right) \right). \end{aligned} \tag{2.15}$$

   Here $\boldsymbol{C}$ is introduced in APIC (Jiang et al., 2015a) for angular momentum compensation.

2. **Grid Update.** Solve $\hat{\boldsymbol{v}}^{n+1}$ using Equation (2.12), where $* = n$ for explicit MPM and $* = n + 1$ for implicit MPM.

3. **Transfer Grid to Particles.** Transfer velocities back to particles and update particle states.

$$\begin{aligned} \boldsymbol{v}_p^{n+1} &= \sum_i \hat{\boldsymbol{v}}_i^{n+1} w_{ip}^n, \\ \boldsymbol{X}_p^{n+1} &= \boldsymbol{X}_p^n + \Delta t \hat{\boldsymbol{v}}_p^{n+1}, \\ \boldsymbol{C}_p^{n+1} &= \frac{4}{\Delta x^2} \sum_i w_{ip}^n \boldsymbol{v}_i^{n+1} \left(\boldsymbol{x}_i^n - \boldsymbol{x}_p^n \right)^T, \\ \boldsymbol{F}_p^{n+1} &= \boldsymbol{F}_p^n + \Delta t \sum_i \hat{\boldsymbol{v}} \nabla w_{ip}^T. \end{aligned} \tag{2.16}$$

Plasticity with MPM In multiplicative finite strain plasticity, the deformation gradient is decomposed as $\boldsymbol{F} = \boldsymbol{F}^E \boldsymbol{F}^P$, where $\boldsymbol{F}^E$ is the elastic deformation and $\boldsymbol{F}^P$ is the plastic deformation. This multiplicative decomposition can be interpreted as two sequential steps in the overall deformation: $\boldsymbol{\Phi} = \boldsymbol{\Phi}^E \circ \boldsymbol{\Phi}^P$. First, the body undergoes plastic deformation, modifying its rest shape. Then, treating this new state as the updated rest configuration, elastic deformation generates stress that pulls the body back toward the new rest configuration. The decomposition is governed by the restriction that $\boldsymbol{F}^E$ always fall within a given elastic region defined by $y(\boldsymbol{\tau}) \leq 0$, where $\tau = \boldsymbol{P}\boldsymbol{F}^\top$ is the Kirchoff stress. Plasticity is particularly convenient in explicit MPM because deformation gradients are tracked directly on particles. If the deformation gradient exceeds the elastic region, it can be projected back onto the yield surface ($y(\boldsymbol{\tau}) = 0$) through a projection map known as plastic return mapping. The specific return mapping varies depending on the plasticity model, such as those for snow,

sand, or foam. In explicit MPM, the only modification needed to incorporate plasticity is a modification to the deformation gradient update step at the end of each time step:

$$\boldsymbol{F}_p^{tr,n} = \boldsymbol{F}_p^n + \Delta t \sum_i \hat{\boldsymbol{v}} \nabla w_{ip}^T, \quad \boldsymbol{F}_p^{n+1} = \mathcal{Z}(\boldsymbol{F}_p^{tr,n}), \tag{2.17}$$

where $\mathcal{Z}$ is a plastic return mapping. From this perspective, the deformation gradients tracked on particles actually correspond to the elastic deformation gradients rather than the total deformation.

2.2.2.3 Extended Position Based Dynamics

For hyperelastic materials, the momentum equation can generally be expressed as follows:

$$\boldsymbol{M}\ddot{\boldsymbol{x}} = -\nabla U(\boldsymbol{x}), \tag{2.18}$$

where, $\boldsymbol{x} = [x_1, x_2, ..., x_p]^T$ is the unknown position states, $\boldsymbol{M}$ is the mass matrix and $U(\boldsymbol{x})$ is the elastic potential energy. eXtended Position-based Dynamics (XPBD) (Macklin et al., 2016) assumes $U(\boldsymbol{x})$ can be further expressed as

$$U(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{C}(\boldsymbol{x})^T \boldsymbol{a}^{-1} \boldsymbol{C}(\boldsymbol{x}), \tag{2.19}$$

where $\boldsymbol{C} = [C_1(\boldsymbol{x}), C_2(\boldsymbol{x}), ..., C_m(\boldsymbol{x})]^T$ consists of m constraints, and $\boldsymbol{a}$ is a diagonal compliance matrix related to the material stiffness. For example, each μ-term and λ-term associated with each mesh element can be treated as a separate constraint. The time stepping balances the equations $\boldsymbol{M}\ddot{\boldsymbol{x}} = 0$ and $\boldsymbol{C}(\boldsymbol{x}) = 0$, leading to the following discrete constrained equations of motion:

$$\boldsymbol{M}(\boldsymbol{x} - \tilde{\boldsymbol{x}}) - \nabla \boldsymbol{C}(\boldsymbol{x})\boldsymbol{\lambda} = 0, \tag{2.20}$$

$$\boldsymbol{C}(\boldsymbol{x}) + \tilde{\boldsymbol{\alpha}}\boldsymbol{\lambda} = 0, \tag{2.21}$$

where $\tilde{\boldsymbol{x}} = \boldsymbol{x}^n + \boldsymbol{v}\Delta t + \boldsymbol{g}\Delta t^2$ represents the predicted position under inertia, $\boldsymbol{\lambda} = -\tilde{\boldsymbol{a}}^{-1}\boldsymbol{C}(\boldsymbol{x})$ is the Lagrange multiplier and $\tilde{\boldsymbol{a}} = \frac{\boldsymbol{a}}{\Delta t^2}$. With certain approximations, the system is solved iteratively using the following incremental equation:

$$\begin{bmatrix} \boldsymbol{M} & -\nabla C \\ -\nabla C & \tilde{\boldsymbol{\alpha}} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{x} \\ \Delta \boldsymbol{\lambda} \end{bmatrix} = - \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{C}(\boldsymbol{x}) + \tilde{\boldsymbol{\alpha}}\boldsymbol{\lambda} \end{bmatrix}, \tag{2.22}$$

where the coefficient matrix and the right-hand side are evaluated using the previous values of $\boldsymbol{x}, \boldsymbol{\lambda}$ and subsequently updated using the computed increments. Although XPBD falls within the umbrella of implicit methods, it does not require expensive Hessian evaluations. Due to its efficiency, it is widely used in real-time game engines.

2.3 Contact Modeling

MPM supports automatic grid-based collision handling; however, it assumes sticky contact between MPM bodies, which may lead to artifacts. In contrast, FEM provides well-defined object boundaries that can be leveraged to define accurate contact mechanics. Incremental Potential Contact (IPC) (Li et al., 2020, 2021a) is a mesh-based collision model that formulates collision forces using smooth log barriers on unsigned distances to ensure separation between objects. It uses smooth log barriers on unsigned distances to ensure separations between objects. It naturally integrates into optimization-based time integrators with an additional energy term, functioning as a "plug-and-play" component. Specifically, collisions between mesh surfaces are classified into point-triangle and edge-edge collisions. The contact potential is defined as:

$$B(x) = \sum_{P,T} b(\mathrm{dist}(P,T)) + \sum_{E_1,E_2} b(\mathrm{dist}(E_1,E_2)), \tag{2.23}$$

where b is a smooth log barrier function of the unsigned distance:

$$b(d) = \begin{cases} -(d-\hat{d})^2 \log(d/\hat{d}), & 0 < d < \hat{d}, \\ 0, & d \geq \hat{d}. \end{cases} \tag{2.24}$$

Here, (P, T) represents an arbitrary point-triangle pair, and (E_1, E_2) represents an arbitrary edge-edge pair. A bounding volume hierarchy (BVH) technique is used to identify all candidate contact pairs efficiently. The parameter $\hat{d}$ controls the size of the contact zone, within which the potential energy increases from zero to infinity as the contact pair approaches each other. During each line search step in optimization-based time integration, continuous collision detection (CCD) is employed to determine an upper bound on the step size. The combination of the log barrier and CCD ensures that penetrations do not occur. Furthermore, the smoothness of the barrier function allows for robust handling of arbitrarily large time step sizes.

This framework also enables precise control over frictional forces by incorporating a locally smoothed semi-implicit Coulomb friction model into the potential energy formulation. For each contact point $\boldsymbol{x}_k$ with sliding basis $\boldsymbol{T}_k$ and normal contact force λ_k, the local friction force is defined as:

$$f_k(\boldsymbol{x}_k) = -\mu\lambda_k f_1(\|\boldsymbol{u}_k\|)\frac{\boldsymbol{u}_k}{\|\boldsymbol{u}_k\|}, \tag{2.25}$$

where μ is the frictional coefficient, $\boldsymbol{u}_k$ is the relative tangential displacement along $\boldsymbol{T}_k$, f_1 is a function smoothly increase from 0 to 1 in the region $[0, \epsilon_v h]$ with ϵ_v controling the region of static friction. By temporarily treating $\boldsymbol{T}_k$ and λ_k as constants, the above friction force can be spatially integrated into a frictional potential for incorporation into optimization-based time integration:

$$b_k^f(\boldsymbol{x}_k) = -\mu\lambda_k f_0(\|\boldsymbol{u}_k\|). \tag{2.26}$$

2.4 Neural Scene Representations

2.4.1 Neural Radiance Fields

Neural Radiance Fields (NeRFs) represent a 3D scene using a view-independent volume density field $\sigma(\mathbf{x})$ and a view-dependent appearance (color) field $\mathbf{c}(\mathbf{x}, \omega)$ for each point $\mathbf{x} \in \mathbb{R}^3$, and view direction $\omega = (\theta, \phi) \in \mathbb{S}^2$ (spherical coordinates). Color images are rendered from these fields by sampling points along a ray for each pixel in the output image. The

appearance $\mathbf{C}(\mathbf{r})$ of a pixel specified by ray direction $\mathbf{r}(s)$ ($s \in [s_{\min}, s_{\max}]$) is given by the volume rendering integral (Mildenhall et al., 2020)

$$\boldsymbol{C}(\boldsymbol{r}) = \int_{s_n}^{s_f} T(s)\ \sigma(\boldsymbol{r}(s))\ \boldsymbol{c}(\boldsymbol{r}(s), \omega)\ ds, \quad T(s) = \exp\left(-\int_{s_n}^{s} \sigma(\boldsymbol{r}(\bar{s}))\ d\bar{s}\right), \tag{2.27}$$

where $T(s)$ is the accumulated transmittance, representing the probability that light travels unoccluded up to depth.

Given a set of ground-truth multiview images, the NeRF model is trained by enforcing the rendered pixel colors to match those in the multiview images. This is achieved by minimizing the rendering loss:

$$\mathcal{L}_{render} = \frac{1}{|\mathcal{R}|} \sum_{\boldsymbol{r} \in \mathcal{R}} \|\boldsymbol{C}(\boldsymbol{r}) - \hat{\boldsymbol{C}}(\boldsymbol{r})\|, \tag{2.28}$$

where $\mathcal{R}$ is the set of sampled rays, $\hat{\boldsymbol{C}}(\boldsymbol{r})$ is the ground truth color observation.

2.4.2 3D Gaussian Splatting

3D Gaussian Splatting (Kerbl et al., 2023) reparameterizes the above NeRF representation using a set of 3D Gaussian kernels $\{\boldsymbol{x}_p, \sigma_p, \boldsymbol{A}_p, \mathcal{C}_p\}_{p \in \mathcal{P}}$, where $\boldsymbol{x}_p$, σ_p, $\boldsymbol{A}_p$, and $\mathcal{C}_p$ represent the centers, opacities, covariance matrices, and spherical harmonic coefficients of the Gaussians, respectively. The color of each pixel is computed using α-blending:

$$\boldsymbol{C} = \sum_{k \in \mathcal{P}} \alpha_k \mathrm{SH}(\boldsymbol{d}_k; \mathcal{C}_k) \prod_{j=1}^{k-1} (1 - \alpha_j), \tag{2.29}$$

where the Gaussian kernels are sorted by z-depth along the camera direction, α_k represents the effective opacity, $\boldsymbol{d}_k$ denotes the view direction from the camera to $\boldsymbol{x}_k$. In practice, rendering is performed by rasterizing Gaussian kernels onto the screen, rather than computing volume integration as in NeRFs. This makes 3DGS significantly faster and more memory-efficient. The 3DGS representation of a scene is trained using a set of multiview images with a rendering loss similar to that used in NeRFs. Additionally, its explicit scene representation makes 3DGS particularly well-suited for scene editing tasks.

CHAPTER 3

Accurate Inelasticity Modeling

3.1 ECI: Energetically Consistent Inelasticity for Optimization Time Integration

3.1.1 Introduction

Since the pioneering work of Terzopoulos and Fleischer (1988), the computer graphics community has observed increasing interests in modeling inelastic deformations governed by elastoplasticity, viscoelasticity, and viscoplasticity. These inelastic mechanical properties govern the behaviors of a wide range of everyday objects. Drawing inspirations from continuum mechanics, computer graphics researchers have successfully modeled and simulated many inelastic materials, ranging from metal, sand, snow and mud to foam, paint and organic tissues.

Inelasticity (mainly elastoplasticity and viscoplasticity) has been widely explored using mesh-based Finite Elements. During inelastic deformation, extreme element distortion and fracture commonly co-exist. Thus, remeshing (O'Brien et al., 2002) and virtual node (Molino et al., 2004; Hegemann et al., 2013) techniques are often applied. More recently, the Material Point Method (MPM) has emerged as a popular alternative for inelastic materials (Jiang et al., 2016) due to its natural support of topologically changing continuum materials.

Despite a large amount of work in modeling inelasticity, a loss of accuracy occurs in almost all existing work. In particular, when implicit time integration schemes are performed, the plastic strain is often treated as a constant, and the real plastic deformation is imagined to happen instantaneously at the beginning or the end of a time step. Such a semi-implicit

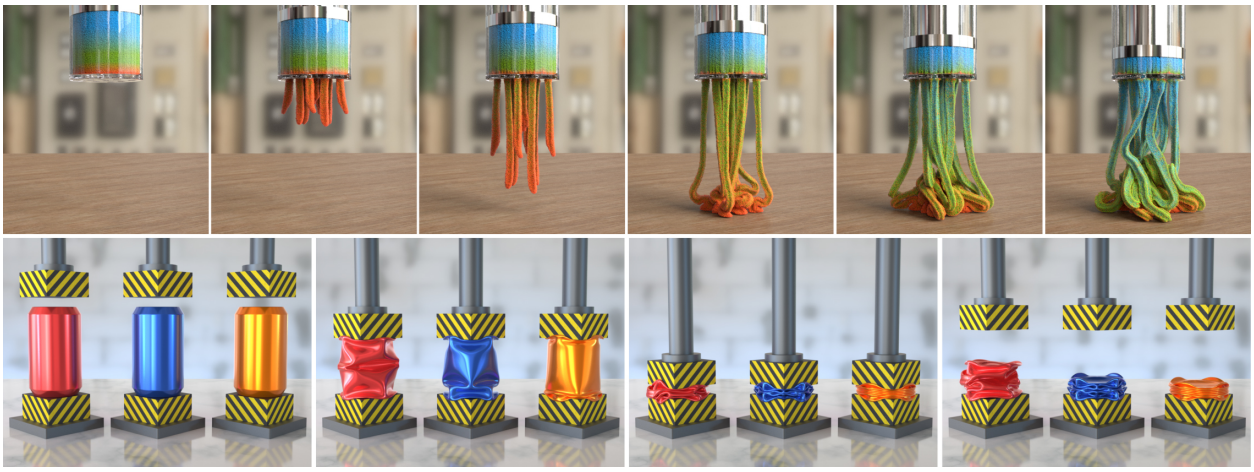


Figure 3.1: Our energetically consistent inelasticity model can not only be applied to the Material Point Method (top row), but also easily extend to the Finite Element Method (bottom row with decreasing hardening coefficients from left to right). The stability under large time steps is guaranteed by the optimization time integration.

lagged treatment of inelasticity results in unnoticeable visual artifacts for certain material models such as the heuristic snow plasticity in Stomakhin et al. (2013) but significant errors such as excessive artificial cohesion for others (Tampubolon et al., 2017; Gao et al., 2018b).

The choice of semi-implicity is largely due to the prominent challenge in modeling implicit inelasticity. Klár et al. (2016) was the first to explore differentiating the plastic flow for Drucker-Prager soil plasticity and incorporating it into the implicit momentum balance. The authors proposed an implicit force formulation that resembles a similar format to semi-implicit formulations (Stomakhin et al., 2013). Unfortunately, their formulation cannot be expressed as the negative gradient of analytical energy. Resultingly, the stiffness matrix is asymmetric, and GMRES became necessary for the associated nonlinear root-finding problem – a problem that by itself has no stability or convergence guarantees when solved with Newton's method. Fang et al. (2019) used alternating direction method of multipliers (ADMM) to shift the asymmetry to local small linear systems, however without an energy, they could not perform global convergence techniques such as line search.

This paper tackles the challenge by revisiting the derivation of implicit plasticity. Our objective is to construct an analytical, augmented potential energy function whose derivative exactly reproduces the implicit force. Related work in classic engineering literature

(Radovitzky and Ortiz, 1999; Ortiz and Stainier, 1999) formulated variational constitutive model updates based on the principle of maximum plastic dissipation and minimizing over the so-called dual inelastic potential. Taking a different path, we derive our method based on constructing a smooth energy that is consistent with existing return mapping-based plasticity treatments (Simo and Hughes, 1998) in explicitly integrated inelasticity simulation systems. As a result, our implicit inelasticity formulation can be directly incorporated into recently advanced optimization time integrators (Gast et al., 2015; Wang et al., 2020a; Li et al., 2020) to enable large time step integration with guaranteed stability, theoretical consistency with return mapping, and a symmetric energy Hessian. Our contributions include:

- An implicit internal force formulation for fully implicit finite strain elastoplasticity;
- A strain energy augmentation method that yields analytically integrable elastoplastic forces and symmetric force derivatives for von Mises J2 plasticity;
- An extension of our model to support strain hardening, pressure-dependent soil plasticity, and rate-dependent viscoelasticity;
- Algorithms for incorporating our model in optimization-based time integrators with the Material Point Method and the Finite Element Method.

We demonstrate our results by simulating a wide range of inelastic materials, including metal, sand, snow, and foam. Our method allows the simulations of inelasticity to enjoy the advantages of guaranteed stability, global convergence, and large time step sizes brought by optimization-based time integrators without suffering from inaccuracy and numerical artifacts from prior work.

3.1.2 Related Work

Inelasticity with FEM Elastoplastic simulation with FEM has been extensively explored by the computer graphics community. O'Brien et al. (2002) used the additive decomposition of strain to separate elastic deformations and plastic deformations and used the von-Mises

yield criterion. However, as Irving et al. (2004) pointed out, this decomposition does not support incompressibility for finite strain. Instead, Irving et al. (2004) used the multiplicative decomposition of deformation gradient with the volume-preserving return mapping algorithm. Our model is based on this decomposition as well. Under this framework, large plastic deformations may make the dynamic system ill-conditioned. To solve this problem, Molino et al. (2004) proposed the virtual node algorithm to allow topology changes when the simulated mesh is severely distorted, and Bargteil et al. (2007) used remeshing technique to maintain a high-quality mesh throughout the simulation. For high-performance simulation, Wojtan and Turk (2008) used frequently remeshed high-resolution surfaces combined with low-resolution interior tetrahedral mesh to resolve thin features near the boundaries. Wojtan et al. (2009) further improved the framework to allow topology changes in inelasticity simulations. These methods introduced extra computational costs or complexities. Instead, we use optimization time integrators to maintain long-time stability and global convergence. Furthermore, Bargteil et al. (2007) proposed a volume-preserving plasticity model incorporating creep and work hardening/softening, which is also followed by Wojtan and Turk (2008). These are important requirements for obtaining physical accuracy, which are all supported by our model as well. Jones et al. (2016b) proposed an examples-based approach for the mesh-based discretization, which search rest shapes on a predefined example manifold. This method is efficient for animation purposes but are less physically accurate.

Inelasticity with MPM Extending the work of Harlow (1964) and Brackbill and Ruppel (1986) on PIC/FLIP, MPM was proposed as a hybrid Lagrangian/Eulerian method for solid mechanics by Sulsky et al. (1994). Since its appearance in the graphics community (Stomakhin et al., 2013; Hegemann et al., 2013), it has attracted a lot of attentions. The most prominent advantage of MPM on modeling inelastic materials is its flexibility in handling extreme deformation and topological changes, which pose significant challenges to Lagrangian mesh-based approaches. Snow plasticity was first simulated by Stomakhin et al. (2013) in a semi-implicit fashion, enforcing thresholds on principal stretches with post-projections. Yue et al. (2015) used the Herschel-Bulkley model of non-Newtonian viscoplastic flow to

approximate foam behaviors. Fei et al. (2019) derived an analytic plastic flow approach for Herschel-Bulkley fluid to simulate compressible, shear-dependent liquids. Daviet and Bertails-Descoubes (2016) modeled the granular materials as compressible viscoplastic fluids combined with the Drucker-Prager yield criterion. Their method suits the granular material simulations well, but follows a different perspective from ours. From the perspective of large strain solid mechanics, Klár et al. (2016) simulated granular continuum using the return mapping algorithm for the Drucker-Prager plasticity. Following Klár et al. (2016), Yue et al. (2018) proposed a hybrid method combining both discrete and continuum treatments to achieve a high level of details with less computational costs. Fang et al. (2019) applied the return mapping approach to handle elastoplasticity and viscoelasticity in an ADMM framework. Except for Klár et al. (2016), these methods all temporally discretize inelasticity in an explicit or semi-implicit way, where the plastic correction was performed as an extra step at the end of each time step, fully decoupled from elasticity. Decoupled treatment in an explicit integration can be justified via operator splitting; however, it will cause artifacts for a (semi-)implicit integration. We use the return mapping framework as well for our fully implicit elastoplasticity and viscoelasticity, and we will show that ours is more temporally consistent compared to Klár et al. (2016).

Inelasticity with Other Discretizations Inelasticity simulations are also explored with other types of spatial discretizations, e.g., Smoothed Particle Hydrodynamics (SPH), Position Based Dynamics (PBD), peridynamics, etc.

SPH is a mesh-free Lagrangian method originally invented for fluid simulations. Inspired by SPH, Müller et al. (2004); Jones et al. (2014) applied the plasticity model in O'Brien et al. (2002) to moving least square particles for elastoplastic objects. Clavet et al. (2005) used springs between particles to mimic elasticity and achieved plasticity by modifying rest lengths during the simulation. These two plasticity models are not derived from the finite strain framework. Alduán and Otaduy (2011) simulated granular materials using an incompressible SPH framework combined with the Drucker-Prager yield criterion. Their plastic correction was performed in a Jacobi-like manner until convergence, while ours is performed with

fixed-point iterations. Yang et al. (2017a) proposed an elastoplastic model based on the Drucker-Prager yield criterion as well within an SPH framework. Gerszewski et al. (2009) introduced deformation gradients to the SPH framework so that plasticity models based on the multiplicative decomposition of deformation gradient can be applied. They used explicit time integrators combined with the plasticity model in (Irving et al., 2004). Gissler et al. (2020) used an implicit compressible SPH solver to simulate the compression of snow. The plasticity is handled by an extra correction step on the deformation gradient following Stomakhin et al. (2013), which is still a semi-implicit method.

PBD was proposed by (Müller et al., 2007) for real-time simulations. This method replaced internal forces in force-based methods with constraints on positions. Plastic deformations can be introduced by the shape matching framework (Müller et al., 2005; Jones et al., 2016a; Falkenstein et al., 2017; Bender et al., 2017). However, this simulation framework sacrifices physical accuracy for better efficiency.

The peridynamic theory is an emerging field in simulations, which was proposed by Silling (2000) to handle discontinuities caused by deformations, such as cracks. It defines pairwise force functions between particles and uses the integration over the interactions from neighboring particles to describe dynamics. He et al. (2017) used the peridynamics framework to simulate elastoplastic materials in a projective dynamics way. They adopted the Drucker-Prager criterion for plasticity. Their solver can also be extended to simulate viscoelasiticity. Chen et al. (2018b) derived a form of force functions based on the isotropic linear elasticity model to simulate elastoplastic materials. They used explicit time integrators and an additive plasticity model.

Optimization Time Integration Large-scale implicit simulation methods usually require solving large systems of nonlinear equations. To solve these systems, the Newton method for root-finding problems is usually adopted, which needs careful tuning of the time step size to ensure convergence. In fact, many of these implicit equations can be integrated to get variational forms, where the equivalent minimization problem can be solved by applying robust optimization techniques. The optimization time integrators have advantages in terms

of long-time stability even when simulating severe deformation with large time step sizes.

Bouaziz et al. (2014) proposed Projective Dynamics (PD), which reformulated the backward Euler time integration for a specific type of material into a local-global alternating solver. Both the local and global steps have simple variational forms that can be solved in a robust and efficient way. This framework was later extended to simulate hyperelastic materials (Liu et al., 2017b), support Laplacian damping (Li et al., 2018), and utilize other time integration schemes (Dinev et al., 2018). Narain et al. (2016) then extended PD to a more general form within the ADMM framework. Brown and Narain (2021) improved the ADMM framework to resolve large rotations. Gast et al. (2015) recast the backward Euler time integration with hyperelastic materials, Rayleigh dampings, and collision penalties as a minimization problem. Li et al. (2019a) and Wang et al. (2020a) explored domain decomposed and hierarchical preconditioning strategies respectively within a quasi-Newton optimization framework for robust and efficient time integration. Wang and Yang (2016) proposed a gradient descent solver for GPUs to accelerate optimization time integrations. Li et al. (2020) proposed Incremental Potential Contact (IPC), a variational form for frictional contacts. Their friction bases are iterated in a similar manner to our iterative yield stresses. IPC is later proven effective for simulating codimensional objects (Li et al., 2021a), rigid bodies (Ferguson et al., 2021a), reduced elastic solids (Lan et al., 2021a), and FEM-MPM coupled domains (Li et al., 2021c), all within the optimization time integration framework. In this paper, we follow Gast et al. (2015) and Li et al. (2020) for the optimization time integration of MPM and FEM respectively. The hessian matrices are enforced to be positive definite following the per-stencil projection technique in Teran et al. (2005).

In addition to hyperelastic solids, Batty et al. (2007) reformulated the classical pressure projection step in solid-fluid coupling as a kinetic energy minimization. Narain et al. (2010) used a hybrid method for sand simulation, where the pressure and friction are solved on the Eulerian grid with a staggered projection method, alternating between two coupled quadratic programs. Narain et al. (2012) posed the strain limiting in cloth simulation as a nonlinear optimization problem. Karamouzas et al. (2017) proposed an energy-based crowd model for crowd simulation. Inglis et al. (2017) formulated fluid simulation under a primal-dual

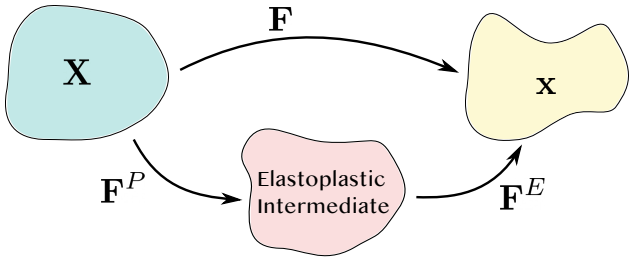


Figure 3.2: The elastoplastic decomposition of the deformation gradient.

optimization framework. Brown et al. (2018) proposed an energy for dissipative forces.

3.1.3 Foundations

In this section we start with reviewing finite strain elastoplasticity (Section 3.1.3.1), MPM spatial discretization (Section 3.1.3.2), optimization-based time integration (Section 3.1.3.3), and discretized plastic flow rule (Section 3.1.3.4). Our review is by no means complete, and they are provided as necessary background knowledge for our new model.

In Section 3.1.3.5, we present a new implicit force formulation that is consistent with the variational weak form. It has a remarkable advantage – integrability, and thus lays important theoretical foundations for our method.

3.1.3.1 Finite Strain Elastoplasticity

Our variational inelasticity model is derived under the finite strain elastoplasticity framework. Here we review some basic concepts and refer to (Simo, 1992; Simo and Hughes, 1998) for more details.

Let $\Omega^0 \subset \mathbb{R}^3$ be the reference configuration of the continuum body and denote $\boldsymbol{x} := \Phi(\boldsymbol{X}, t)$ the deformation map from Ω^0 (with coordinate $\boldsymbol{X}$) to the world space Ω^t (with coordinate $\boldsymbol{x}$). The deformation gradient $\boldsymbol{F} = \frac{\partial \Phi}{\partial \boldsymbol{X}}(\boldsymbol{X}, t)$ measures the local deformation of the infinitesimal region around $\boldsymbol{X}$. With finite strain elastoplasticity, $\boldsymbol{F}$ is multiplicatively decomposed into $\boldsymbol{F} = \boldsymbol{F}^E \boldsymbol{F}^P$, where $\boldsymbol{F}^P$ denotes the permanent plastic deformation, and $\boldsymbol{F}^E$ denotes the elastic deformation which results in elastic forces (Figure 3.2). Plasticity requires that the Kirchhoff stress $\boldsymbol{\tau}$ associated with $\boldsymbol{F}^E$ is inside the admissible area defined by a yield condition

$y(\boldsymbol{\tau}) \leq 0$. The surface characterized by $y(\boldsymbol{\tau}) = 0$ is often referred to as the yield surface. When $\boldsymbol{F}$ changes, $\boldsymbol{F}^E$ will follow some plastic flow to evolve so that it lies within the yield surface. In this paper, we follow the volume preserving plastic flow from (Klár et al., 2016).

From the Lagrangian view point, the state of dynamics of an elastoplastic continuum can be described by a Lagrangian density field $R(\boldsymbol{X}, t)$ on Ω^0 and a Lagrangian velocity field $\boldsymbol{V}(\boldsymbol{X}, t) := \frac{\partial \Phi(\boldsymbol{X}, t)}{\partial t}$ on Ω^0. The two fields are governed by the conservation of mass

$$R(\boldsymbol{X}, t)J(\boldsymbol{X}, t) = R(\boldsymbol{X}, 0), \tag{3.1}$$

where $J = \det \boldsymbol{F}$, and the conservation of momentum,

$$R(\boldsymbol{X}, 0)\frac{\partial \boldsymbol{V}}{\partial t}(\boldsymbol{X}, t) = \nabla^{\boldsymbol{X}} \cdot \boldsymbol{P} + R(\boldsymbol{X}, 0)\boldsymbol{g}, \tag{3.2}$$

where $\boldsymbol{P}$ is the first Piola-Kirchhoff stress and $\boldsymbol{g}$ is the gravity.

3.1.3.2 MPM Discretization

The Material Point Method (MPM) discretizes a continuum by a set of disconnected Lagrangian material particles. The continuous time variable t is discretized by consecutive time steps $t^0, t^1, ..., t^n$. Without loss of generality, we assume a fixed time step size Δt. The advection is carried on material particles, so the conservation of mass across time steps is trivially satisfied. Assuming backward difference on $\frac{\partial \boldsymbol{V}}{\partial t}$, the weak form of the momentum equation is (Jiang et al., 2016)

$$\frac{1}{\Delta t}\int_{\Omega^0} R(\boldsymbol{X}, 0)(\boldsymbol{V}^{n+1} - \boldsymbol{V}^n)Q_\alpha d\boldsymbol{X} = -\int_{\Omega^0} \boldsymbol{P}\nabla^{\boldsymbol{X}} Q_\alpha d\boldsymbol{X} \tag{3.3}$$

for an arbitrary test function $Q_\alpha(\boldsymbol{X}, t^n)$.

Here for simplicity, we drop the gravity term and admit the free surface assumption.

MPM uses the previous time step t^n as the reference configuration, so the integration on

Ω^0 is pushed forward onto Ω^n:

$$\frac{1}{\Delta t}\int_{\Omega^n}\rho(\boldsymbol{x},t^n)(\hat{\mathbf{v}}^{n+1}-\mathbf{v}^n)q_\alpha d\boldsymbol{x} = -\int_{\Omega^n}\frac{1}{J^n}\boldsymbol{P}F^{nT}\nabla^{\boldsymbol{x}}q_\alpha d\boldsymbol{x}, \tag{3.4}$$

where $\rho, \mathbf{v}^n, \hat{\mathbf{v}}^{n+1}$, and q_α are Eulerian counterpart of $\mathcal{R}, \mathbf{V}^n, \mathbf{V}^{n+1}$, and Q_α, obtained by pushing forward from Ω^0 onto Ω^n.

In MPM, B-Spline-based interpolations are often applied to define fields on Ω^n, and material particles serve as quadratures to approximate the volume integration. Let $x_p^n, \boldsymbol{X}_p$ be the coordinate of particle p in Ω^n and Ω^0 respectively, and w_{ip}^n and ∇w_{ip}^n be the weight and weight gradient between particle p and grid i. With the mass lumping technique, the force equilibrium of grid i can be discretized as

$$\frac{1}{\Delta t}m_i^n(\hat{\mathbf{v}}_i^{n+1}-\mathbf{v}_i^n) = -\sum_p \boldsymbol{P}_p\boldsymbol{F}_p^{nT}\nabla w_{ip}^n V_p^0, \tag{3.5}$$

where V_p^0 is the initial volume of particle p, $m_i^n = \sum_p m_p w_{ip}^n$ is the lumped mass on grid i and m_p is approximated by $R(\boldsymbol{X}_p, 0)V_p^0$. The right hand side of Equation Equation (3.5) is the internal elastic force on grid i.

At each time step, the velocity field $\mathbf{v}^n$ is transferred from material particles to grid nodes, and the new velocity field $\hat{\mathbf{v}}^{n+1}$ is solved and transferred back to material particles for advection. In this paper, we use the quadratic MLS kernel (Hu et al., 2018a) as the weight function, and APIC (Jiang et al., 2015b) as the particle-grid transfer scheme.

3.1.3.3 Optimization Time Integration

Assuming implicit integration with BDF1 (backward Euler), the first Piola-Kirchhoff stress $\boldsymbol{P}$ in Equation Equation (3.5) is associated with the deformation gradient $\boldsymbol{F}^{n+1}$ at time step t^{n+1}. The deformation gradients $\boldsymbol{F}^n$ and $\boldsymbol{F}^{n+1}$ are related by

$$\boldsymbol{F}^{n+1} = (\mathbf{I} + \Delta t\nabla\hat{\mathbf{v}}_p^{n+1})\boldsymbol{F}^n, \tag{3.6}$$

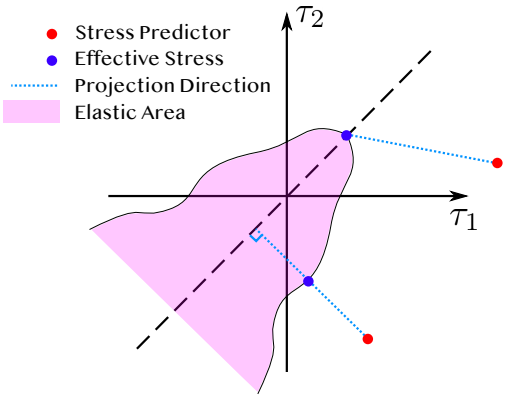


Figure 3.3: Return mapping for a discrete plastic flow.

where $\nabla\hat{\mathbf{v}}_p^{n+1} = \sum_i \hat{\mathbf{v}}_i^{n+1}\nabla w_{ip}^{n\,\top}$.

Existing optimization-based time integrators in computer graphics often assume hyperelastic materials. Without plasticity, the first Piola-Kirchhoff stress is simply the derivative of the corresponding elastic strain energy density function: $\boldsymbol{P}(\boldsymbol{F}) = \frac{\partial\Psi}{\partial\boldsymbol{F}}$. Equation Equation (3.5) is then equivalent to the following optimization problem:

$$\begin{aligned}\Delta\hat{\mathbf{v}} &= \text{argmin}_{\Delta\mathbf{v}}\, E(\Delta\mathbf{v}) = \sum_i m_i\|\Delta\mathbf{v}_i\|^2 + \sum_p \Psi(\boldsymbol{F}_p^{tr}(\mathbf{v}^n + \Delta\mathbf{v}))V_p^0,\\ \hat{\mathbf{v}}^{n+1} &= \mathbf{v}^n + \Delta\hat{\mathbf{v}},\end{aligned} \tag{3.7}$$

where $\boldsymbol{F}_p^{tr}(\mathbf{v}) = (\mathbf{I} + \Delta t\nabla\mathbf{v}_p)\boldsymbol{F}_p^n$ is the elastic predictor. The optimization problem can be robustly solved by projected Newton's method with backtracking line search (Wang et al., 2020a).

Gravity. For the effect of gravity, we add the term $m_i\boldsymbol{g}$ on the right-hand side in Equation (3.5), which corresponds to the extra term $-\sum_i m_i\boldsymbol{g}^\top\mathbf{v}$ in Equation (3.7).

3.1.3.4 Discretization of Plastic Flow

In the discrete setting, plasticity is most commonly achieved by the return mapping algorithm (Simo and Hughes, 1998; Klár et al., 2016), which is equivalent to solving for a strain that satisfies the plastic flow rule. Geometrically, the return mapping defines how elastic predictors

outside the yield surface should be corrected so that the effective stresses lie inside the yield surface. We follow the notations in Klár et al. (2016) to describe discrete plastic flows in this paper. For elasticity, we adopt the St. Venant-Kirchhoff (StVK) model with Hencky strains. The elastoplasticity of isotropic materials can be characterized in the principal stretch space using the singular value decomposition (SVD) (Stomakhin et al., 2012). Let $\boldsymbol{F}^{tr} = \boldsymbol{U}\boldsymbol{\Sigma}^{tr}\boldsymbol{V}^\top$ be the SVD of an elastic predictor $\boldsymbol{F}^{tr}$. The Hencky strain is defined as $\boldsymbol{\epsilon} = \log \boldsymbol{\Sigma}^{tr}$, and the Kirchhoff stress for the StVK model is $\boldsymbol{\tau} = 2\mu\boldsymbol{\epsilon} + \lambda \operatorname{tr}(\boldsymbol{\epsilon})\boldsymbol{I}$, where μ, λ are Lamé parameters. For a discrete plastic flow, if the stress associated with an elastic predictor is outside the yield surface, then the stress is projected back onto the yield surface. The projection procedure in the principal stress space is illustrated in Figure 3.3. Note that along the perpendicular direction to the diagonal, we would have $\det \boldsymbol{F}^P = 1$, which corresponds to a volume-preserving plastic deformation. We denote the endpoint of the return mapping (also known as the corrector or the effective stress) as $\boldsymbol{F}^E = \mathcal{Z}(\boldsymbol{F}^{tr})$ where $\mathcal{Z}(\cdot)$ is the return mapping.

3.1.3.5 Force Balance with Implicit Plasticity

For elastoplastic materials, the first Piola-Kirchhoff stress $\boldsymbol{P}$ in Equation (3.5) should be rewritten as (Bonet and Wood, 1997)

$$\boldsymbol{P} = \frac{\partial \Psi^E}{\partial \boldsymbol{F}^E} \boldsymbol{F}^{P-\top}. \tag{3.8}$$

Here Ψ^E is the elastic strain energy density function. We add a superscript to emphasize the elastic energy is only associated with the elastic deformation gradient $\boldsymbol{F}^E$.

Through a weak form derivation of the updated Lagrangian dynamics (see the supplemental document for details), we show that the implicit internal force on grid node i is:

$$\boldsymbol{f}_i^{n+1} = -\sum_p V_p^0 \frac{\partial \Psi^E}{\partial \boldsymbol{F}^E}(\boldsymbol{F}_p^{E,n+1}) \boldsymbol{F}_p^{P,n+1-\top} \boldsymbol{F}_p^{n\top} \nabla w_{ip}^n, \tag{3.9}$$

In practice one does not need to track the plastic deformation gradients $\boldsymbol{F}^P$ on the particles.

The nodal force can be expressed in terms of $\boldsymbol{F}^{E,tr}$ (defined as $\boldsymbol{F}^{E,tr}(\mathbf{v}) = (\mathbf{I} + \Delta t \nabla \mathbf{v}_p)\boldsymbol{F}_p^{E,n}$):

$$\boldsymbol{f}_i^{n+1} = -\sum_p V_p^0 \frac{\partial \Psi^E}{\partial \boldsymbol{F}^E}(\mathcal{Z}(\boldsymbol{F}^{E,tr}))\mathcal{Z}(\boldsymbol{F}^{E,tr})^\top \boldsymbol{F}^{E,tr^{-\top}} \boldsymbol{F}^{E,n^\top} \nabla w_{ip}^n. \tag{3.10}$$

Note that when doing explicit time integration, we can directly replace $\boldsymbol{F}_p^{E,n+1}, \boldsymbol{F}_p^{E,tr}$ both with $\boldsymbol{F}_p^{E,n}$, which gives the common force expression for explicit MPM:

$$\boldsymbol{f}_i^n = -\sum_p V_p^0 \frac{\partial \Psi^E}{\partial \boldsymbol{F}^E}(\boldsymbol{F}_p^{E,n})\boldsymbol{F}^{E,n^\top} \nabla w_{ip}^n. \tag{3.11}$$

For implicit plasticity, Klár et al. (2016) directly replace $\frac{\partial \Psi^E}{\partial \boldsymbol{F}^E}(\boldsymbol{F}_p^{E,n})$ in Equation (3.11) with $\frac{\partial \Psi^E}{\partial \boldsymbol{F}^E}(\mathcal{Z}(\boldsymbol{F}_p^{E,tr}))$ and define the resulting expression as the implicit force. We can clearly observe that

$$\boldsymbol{f}_i^{n+1} \neq -\sum_p V_p^0 \frac{\partial \Psi^E}{\partial \boldsymbol{F}^E}(\mathcal{Z}(\boldsymbol{F}_p^{E,tr}))\boldsymbol{F}^{E,n^\top} \nabla w_{ip}^n, \tag{3.12}$$

i.e., the implicit force in (Klár et al., 2016) is not equivalent to our formulation (Equation (3.10)). As we discuss in the supplemental document, the choice of Klár et al. (2016) is only semi-implicit. Furthermore, their formulation is not integrable because their force derivative is not symmetric. Therefore in (Klár et al., 2016) it is necessary to adopt Newton-Ralphson root finding with GMRES for the asymmetric linear system solve. In the next section, we elaborate on our new model which enables the existence of an analytical energy.

3.1.4 Energetically Consistent Inelasticity (ECI)

3.1.4.1 One-Dimensional Investigation

To motivate ECI, let's start with applying a discrete plasticity model to a one-dimensional spring with a constant yield stress.

Consider a one-dimensional elastoplastic spring with rest length $V_0 = 1$. We fix its one end at $x = 0$, and place the other end at 1 initially. With the initial state being the reference configuration, we model the spring with finite strain elastoplasticity, where the deformation

gradient can be conveniently calculated as $F(x) = x$ with x being the coordinate of its free end.

Discretizing time into steps $t^0, t^1, ..., t^n$ with equal time step size Δt, for time step n, the elastic predictor $F^{E,tr}$ by assuming a purely elastic deformation is given by

$$F^{E,tr}(x) = \frac{1}{F^{P,n}}x. \tag{3.13}$$

We assign the following strain energy density function:

$$\Psi^E(F^E) = \frac{k}{2}(\log F^E)^2, \tag{3.14}$$

where k is the stiffness. Viewing the spring as a single-element FEM discretization, since $V_0 = 1$, Ψ^E equals the total elastic potential. Assume $k = 1$ for brevity, the Kirchhoff stress is then given by

$$\tau(F^E) := \frac{\partial \Psi^E}{\partial F^E}F^E = \log(F^E). \tag{3.15}$$

Let the constant yield stress be $\tau_Y = \log(F_Y)$, where $F_Y \in [1, \infty)$ is a critical strain, and define the yield function to be $\tau - \tau_Y \leq 0$, we can then follow standard plasticity treatment (Simo and Hughes, 1998) to define a simple return mapping procedure with the form

$$F^{E,n+1} = \mathcal{Z}(F^{E,tr}) = \begin{cases} F^{E,tr} & F^{E,tr} \leq F_Y \\ F_Y & \text{otherwise} \end{cases}. \tag{3.16}$$

In terms of the Hencky strain $\epsilon^E = \log(F^E)$, the yield condition is equivalent to

$$\epsilon^{E,tr} - \epsilon^{E,n+1} = \delta\gamma > 0. \tag{3.17}$$

Geometrically, the quantity $\delta\gamma$ measures how far away the elastic strain predictor is from the yield surface in the principal strain space. This quantity plays an important role in

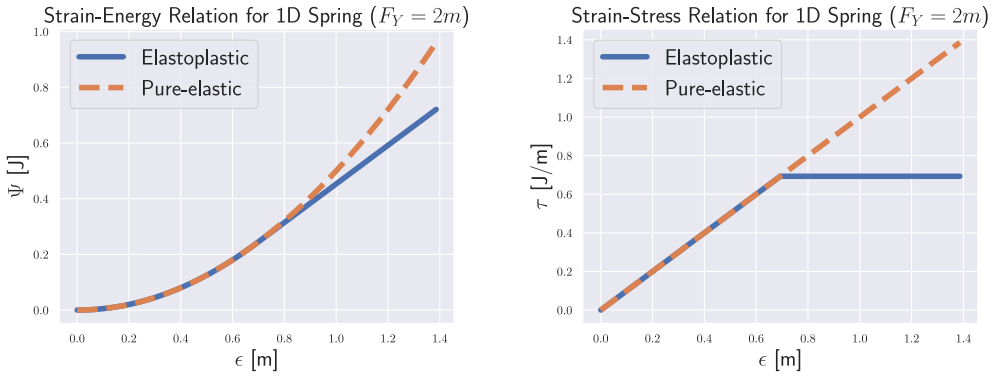


Figure 3.4: The strain-energy (left) and the strain-stress (right) plot of an elastoplastic spring. Here $\epsilon = \log(F)$ and $\tau = \frac{\partial \Psi}{\partial F} F$.

our variational modeling of plasticity. Specifically, we have the following theorem for the elastoplastic springs:

Theorem 3.1.1 (Augmented energy density for springs). In the problem setting described above ($V^0 = 1$, $k = 1$), using the following energy density function

$$\boxed{\Psi(x) = \begin{cases} \Psi^E(\mathcal{Z}(F^{E,tr}(x))) + \tau_Y \delta\gamma(F^{E,tr}(x)) & F^{E,tr} > F_Y \\ \Psi^E(F^{E,tr}(x)) & \text{otherwise} \end{cases}} \tag{3.18}$$

reveals a force that is equivalent to what one would get if one performed the force-based implicit plasticity.

We include in Appendix A the proof for this theorem as well as details showing that $\Psi(x)$ is piecewise C^∞ and everywhere C^1.

A comparison between the augmented energy and the pure-elastic energy is shown in Figure 3.4.

Taking the inertia into consideration, we test the model on a small dynamic mass-spring system. At the end of each time step, F_p is updated from the following relation:

$$F^{n+1} = x^{n+1} = \mathcal{Z}(F^{E,tr}) F^{P,n+1} = F^{E,tr} F^{P,n}. \tag{3.19}$$

The ECI simulation results quantitatively match the results using explicit integration (see

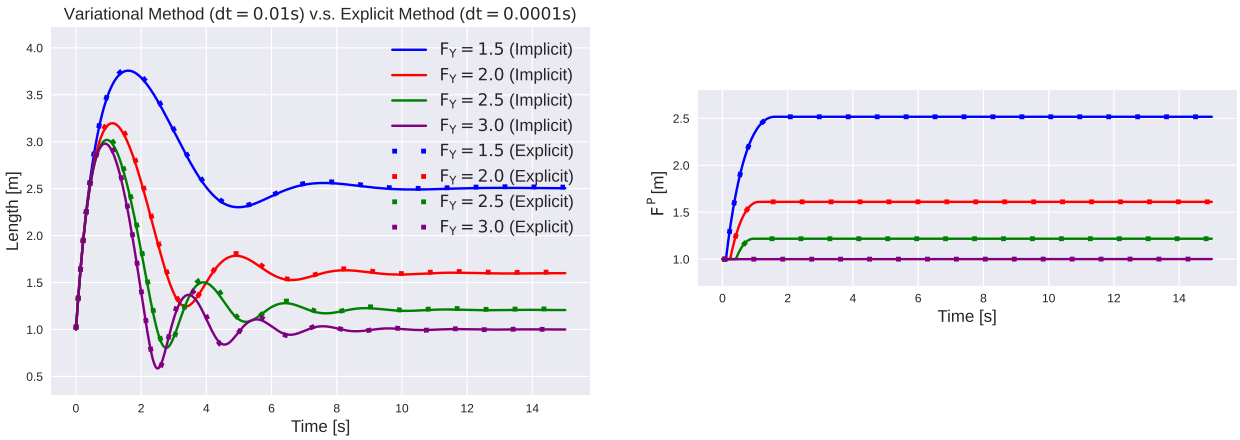


Figure 3.5: **Spring simulation with ECI.** We simulate under the same initial velocity but different critical strains. The results from our large-time-step ECI all match with the results from explicit small-time-step time integration.

Figure 3.5).

3.1.4.2 Extending to Von-Mises Plasticity

A natural analogy of elastoplastic spring with constant yield stress for the plasticity of isotropic hyperelastic materials is the von-Mises plasticity model, which also associates all stress predictors with a constant yield stress τ_Y (the norm of the deviatoric Kirchhoff stress on the yield surface). We study von-Mises plasticity under the St. Venant-Kirchhoff constitutive model with Hencky strains. Following the notations from Section 3.1.3.4, the yield surface is defined as

$$y(\boldsymbol{\tau}) = \|\hat{\boldsymbol{\tau}}\|_F - \tau_Y = 0, \tag{3.20}$$

where $\hat{\boldsymbol{\tau}} = \boldsymbol{\tau} - \frac{1}{d}\operatorname{tr}(\boldsymbol{\tau})\boldsymbol{I}$ is the deviatoric part of the Kirchhoff stress.

The equivalent yield condition is

$$\delta\gamma = \|\hat{\boldsymbol{\epsilon}}\| - \frac{\tau_Y}{2\mu} > 0, \tag{3.21}$$

where $\boldsymbol{\epsilon} = \log(\boldsymbol{\Sigma}^{tr})$ is the trial Hencky strain and $\hat{\boldsymbol{\epsilon}} = \boldsymbol{\epsilon} - \frac{1}{d}\operatorname{tr}(\boldsymbol{\epsilon})\boldsymbol{I}$ is the deviatoric part of

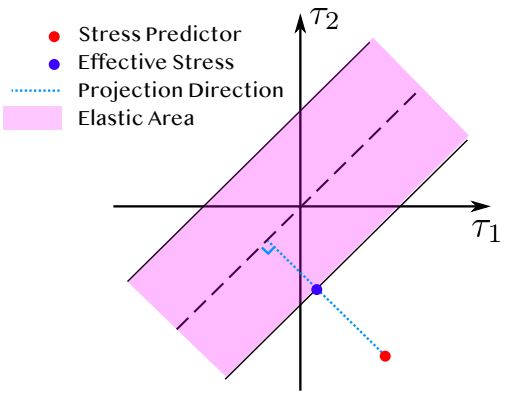


Figure 3.6: Yield surface of the **von-Mises plasticity** model.

the Hencky strain. The corresponding return mapping (Figure 3.6) is

$$\mathcal{Z}(\boldsymbol{F}^{E,tr}) = \begin{cases} \boldsymbol{F}^{E,tr} & \delta\gamma \leq 0 \\ \boldsymbol{U} \exp\left(\boldsymbol{\epsilon} - \delta\gamma \frac{\hat{\boldsymbol{\epsilon}}}{\|\hat{\boldsymbol{\epsilon}}\|}\right) \boldsymbol{V}^\top & \text{otherwise} \end{cases}. \tag{3.22}$$

We have the following key lemma for the von-Mises plasticity:

Lemma 3.1.2. Define the **augmented elastoplastic energy density function** as:

$$\boxed{\Psi(\boldsymbol{F}) = \begin{cases} \Psi^E(\boldsymbol{F}), & \delta\gamma(\boldsymbol{F}) \leq 0 \\ \Psi^E(\mathcal{Z}(\boldsymbol{F})) + \tau_Y \delta\gamma(\boldsymbol{F}), & otherwise \end{cases}}. \tag{3.23}$$

This energy density function satisfies the following identity for any $\boldsymbol{F}$:

$$\frac{\partial \Psi(\boldsymbol{F})}{\partial \boldsymbol{F}} \equiv \frac{\partial \Psi^E}{\partial \boldsymbol{F}^E}(\mathcal{Z}(\boldsymbol{F}))\mathcal{Z}(\boldsymbol{F})^\top \boldsymbol{F}^{-\top}. \tag{3.24}$$

The proof of this lemma is provided in the supplementary document. With this lemma, it is easy to prove the following theorem:

Theorem 3.1.3 (Augmented energy theorem for von-Mises plasticity). The augmented elastoplastic energy density function (Equation (3.23)) viewed as a hyperelastic strain energy density function reveals a force that is equivalent to what one would get if one performed the

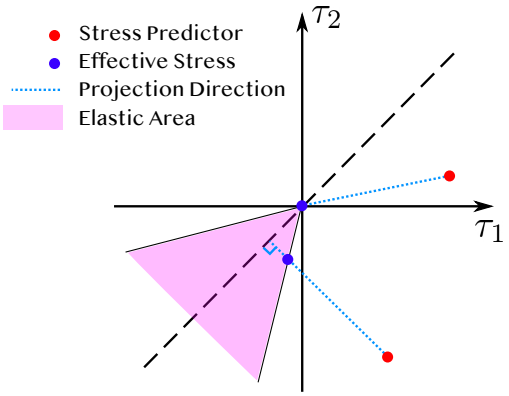


Figure 3.7: Yield surface of the **Drucker-Prager plasticity** model.

force-based implicit plasticity, i.e.

$$\begin{aligned} \boldsymbol{f}_i &= -\sum_p V_p^0 \left[\frac{\partial \Psi}{\partial \boldsymbol{F}}(\boldsymbol{F}_p^{E,tr}) \right] \boldsymbol{F}_p^{E,n\top} \nabla w_{ip}^n \\ &= -\sum_p V_p^0 \left[\frac{\partial \Psi^E}{\partial \boldsymbol{F}_p^E}(\mathcal{Z}(\boldsymbol{F}_p^{E,tr}))\mathcal{Z}(\boldsymbol{F}_p^{E,tr})^\top \boldsymbol{F}_p^{E,tr-\top} \right] \boldsymbol{F}_p^{E,n\top} \nabla w_{ip}^n. \end{aligned} \tag{3.25}$$

When performing optimization time integration, we can simply view the elastoplastic free energy density as a new strain energy density function. At the end of each time step, we update $\boldsymbol{F}^E$ with $\mathcal{Z}(\boldsymbol{F}^{E,tr})$. The detailed pipeline is elaborated in Section 3.1.5.

3.1.4.3 Extending to Pressure Dependent Soil Plasticity

Drucker-Prager plasticity is widely applicable to the simulations of granular materials such as sand. The yield surface under the St. Venant-Kirchhoff constitutive model with Hencky strains is defined as

$$y(\boldsymbol{\tau}) = \|\hat{\boldsymbol{\tau}}\|_F + \alpha \operatorname{tr}(\boldsymbol{\tau}) = 0, \tag{3.26}$$

where $\alpha = \sqrt{\frac{2}{3}} \frac{2\sin\phi_f}{3-\sin\phi_f}$ and ϕ_f is the friction angle.

The equivalent yield condition is then

$$\operatorname{tr}(\boldsymbol{\epsilon}) > 0, \quad \text{or} \quad \delta\gamma = \|\hat{\boldsymbol{\epsilon}}\|_F + \alpha \frac{(d\lambda + 2\mu)\operatorname{tr}(\boldsymbol{\epsilon})}{2\mu} > 0. \tag{3.27}$$

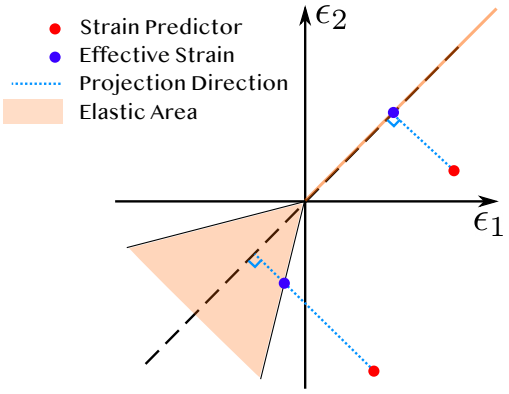


Figure 3.8: Yield surface of the **Drucker-Prager plasticity** model in the principal strain space under our extrapolated St. Venant-Kirchhoff model.

The corresponding return mapping (Figure 3.7) is

$$\mathcal{Z}(\boldsymbol{F}^{E,tr}) = \begin{cases} \boldsymbol{U}\boldsymbol{V}^\top & \mathrm{tr}(\boldsymbol{\epsilon}) > 0 \\ \boldsymbol{F}^{E,tr} & \delta\gamma \leq 0, \mathrm{tr}(\boldsymbol{\epsilon}) \leq 0 \cdot \\ \boldsymbol{U}\exp\left(\boldsymbol{\epsilon} - \delta\gamma\frac{\hat{\boldsymbol{\epsilon}}}{\|\hat{\boldsymbol{\epsilon}}\|}\right)\boldsymbol{V}^\top & \text{otherwise} \end{cases} \tag{3.28}$$

The augmented elastoplastic energy introduced above for our 1D spring and von-Mises model essentially comes from the integrability of the following vector field over $\mathbb{R}^{d\times d}$:

$$\frac{\partial \Psi^E}{\partial \boldsymbol{F}^E}(\mathcal{Z}(\boldsymbol{F}))\mathcal{Z}(\boldsymbol{F})^\top \boldsymbol{F}^{-\top}. \tag{3.29}$$

Unfortunately, this integrability does not hold anymore for the Drucker-Prager return mapping. It can be checked that the Jacobian field of the above vector field is not symmetric. Even worse, $\delta\gamma$ is undefined in the region with $\mathrm{tr}(\boldsymbol{\epsilon}) > 0$, because the projection there is not volume-preserving.

3.1.4.4 Extrapolating St. Venant-Kirchhoff

To solve the issue of $\delta\gamma$ for the area defined by $\text{tr}(\boldsymbol{\epsilon}) > 0$, we extrapolate the St. Venant-Kirchhoff constitutive model in this area as:

$$\hat{\Psi}^E(\boldsymbol{\Sigma}) = \begin{cases} \mu\|\hat{\boldsymbol{\epsilon}}\|^2 & \text{tr}(\boldsymbol{\epsilon}) \geq 0 \\ \mu\|\hat{\boldsymbol{\epsilon}}\|^2 + (\frac{\lambda}{2} + \frac{\mu}{d})(\text{tr}(\boldsymbol{\epsilon}))^2 & \text{tr}(\boldsymbol{\epsilon}) < 0 \end{cases}. \tag{3.30}$$

When $\text{tr}(\boldsymbol{\epsilon}) < 0$, $\hat{\Psi}^E$ is just equivalent to the St. Venant-Kirchhoff strain energy density, which separates the deviatoric term and the pressure term. When $\text{tr}(\boldsymbol{\epsilon}) \geq 0$, we extrapolate the energy only with the deviatoric term and define the yield stress to be zero. This extrapolation does not change the yield surface in the principal stress space. Instead, the yield surface in the principal strain space is extended to include the diagonal line of the first quadrant, and all the points on this ray correspond to the tip of the yield surface in the principal stress space (see Figure 3.8). With this extrapolated model, the volume-preserving projection can be done as well in the area of $\text{tr}(\boldsymbol{\epsilon}) \geq 0$, and $\delta\gamma$ is well-defined.

In summary, with our extrapolation, the return mapping is simplified as

$$\mathcal{Z}(\boldsymbol{F}^{E,tr}) = \begin{cases} \boldsymbol{F}^{E,tr}, & \delta\gamma \leq 0 \\ \boldsymbol{U}\exp(\boldsymbol{\epsilon} - \delta\gamma\frac{\hat{\boldsymbol{\epsilon}}}{\|\hat{\boldsymbol{\epsilon}}\|})\boldsymbol{V}^\top, & otherwise \end{cases}, \tag{3.31}$$

where

$$\delta\gamma = \begin{cases} \|\hat{\boldsymbol{\epsilon}}\|, & \text{tr}(\boldsymbol{\epsilon}) > 0 \\ \|\hat{\boldsymbol{\epsilon}}\| + \alpha\frac{d\lambda+2\mu}{2\mu}\,\text{tr}(\boldsymbol{\epsilon}), & \text{otherwise} \end{cases}. \tag{3.32}$$

3.1.4.5 Recover Integrability

To resolve the non-integrability, we update the yield stress iteratively during integration (Figure 3.9). At each time step, we solve a series of optimization problems with constant yield stresses. The yield stress $\tau_{Y,p}^{tr}$ for each particle p is computed from its elastic predictor

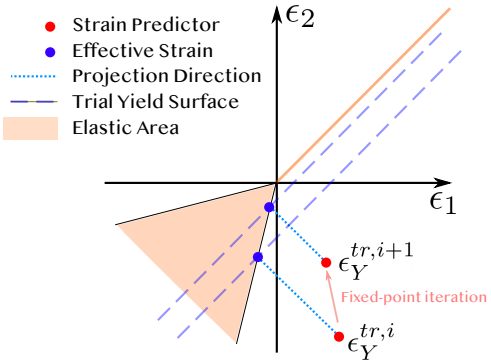


Figure 3.9: Illustration of our iterative stress method for the Drucker-Prager plasticity. i stands for the fixed-point iteration index. The strains here are strain predictors at the beginning of each stress iteration. The trial yield surfaces remain constant within each iteration.

$\boldsymbol{F}_p^{E,tr}$ at the beginning of the optimization:

$$\tau_Y^{tr} = \begin{cases} 0, & \operatorname{tr}(\boldsymbol{\epsilon}) > 0 \\ -\alpha(d\lambda + 2\mu)\operatorname{tr}(\boldsymbol{\epsilon}), & \text{otherwise} \end{cases}, \tag{3.33}$$

and the corresponding $\delta\gamma$ is defined with a fixed yield stress:

$$\delta\gamma = \begin{cases} \|\hat{\boldsymbol{\epsilon}}\|, & \operatorname{tr}(\boldsymbol{\epsilon}) > 0 \\ \|\hat{\boldsymbol{\epsilon}}\| - \frac{\tau_Y^{tr}}{2\mu}, & \text{otherwise} \end{cases}. \tag{3.34}$$

In this way, each particle experiences a local cylinder-like yield surface with a different yield stress. The stress iteration can be viewed as a fixed-point iteration on the yield stresses; see Section 3.1.5.1 for more details. Under convergence, the trial yield stress is consistent with the yield stress defined by the Drucker-Prager yield surface.

3.1.4.6 Drucker-Prager Plasticity with Cohesion.

To simulate materials with both granular and chunky behaviours such as wet sand and snow, we shift the yield surface of Druger-Prager model along the diagonal in the principal stress

space to model cohesion. This effectively updates Equation (3.33) and Equation (3.30) as

$$\tau_Y^{tr} = \begin{cases} 0, & \text{tr}(\boldsymbol{\epsilon}) > cd \\ -\alpha(d\lambda + 2\mu)(\text{tr}(\boldsymbol{\epsilon}) - cd), & otherwise \end{cases}, \tag{3.35}$$

$$\hat{\Psi}^E(\boldsymbol{\Sigma}) = \begin{cases} \mu\|\hat{\boldsymbol{\epsilon}}\|^2 + (\frac{\lambda}{2} + \frac{\mu}{d})(cd)^2 & \text{tr}(\boldsymbol{\epsilon}) \geq cd \\ \mu\|\hat{\boldsymbol{\epsilon}}\|^2 + (\frac{\lambda}{2} + \frac{\mu}{d})(\text{tr}(\boldsymbol{\epsilon}))^2 & \text{otherwise} \end{cases}, \tag{3.36}$$

where $c > 0$ is the cohesion parameter.

3.1.4.7 Hardening

The hardening mechanism plays an important role in simulations of materials like metal (Chakrabarty and Drugan, 1988) and snow (Stomakhin et al., 2013; Gaume et al., 2018). In general, the hardening mechanism is associated with some hardening state set q_n and some hardening parameter set ξ. Theoretically, hardening controls how the yield surface evolves according to the hardening state.

A linear hardening rule for the von-Mises plasticity can be defined as

$$\begin{aligned} q^{n+1} &= q^n + 2\mu\xi\delta\gamma(\boldsymbol{F}^{E,tr}), \\ \tau_Y^{n+1} &= q^{n+1}, \\ q^0 &= \tau_{Y,\text{init}}. \end{aligned} \tag{3.37}$$

This effectively makes the yield stress τ_Y^{n+1} in the equilibrium state at time step t^n depend on $\boldsymbol{F}^{E,tr}$, which is not a constant anymore for different $\boldsymbol{F}^{E,tr}$. Similarly to the iterative stress update for Drucker-Prager, we can also iterate on the hardening state. At the beginning of each optimization, the trial hardening state and the trial yield stress are updated as

$$\tau_Y^{tr} = q^{tr} = q^n + \xi\delta\gamma(\boldsymbol{F}^{E,tr}). \tag{3.38}$$

At the end of the time step, the hardening state q^{n+1} is updated to be the last trial hardening

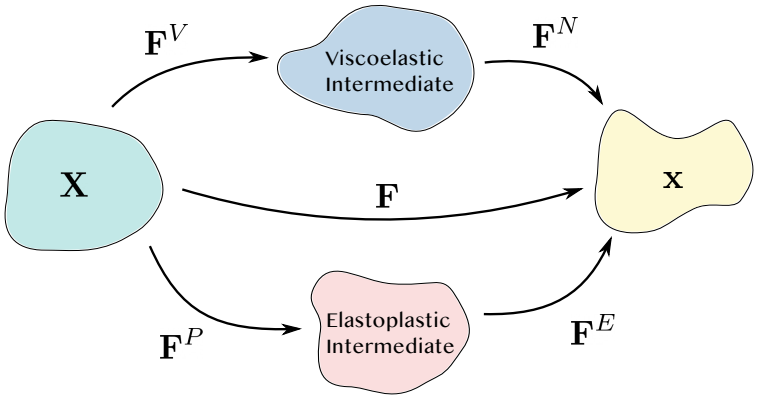


Figure 3.10: The viscoelastic decomposition of the deformation gradient and its relationship to the elastoplastic decomposition.

state q^{tr}.

3.1.4.8 Viscoelasticity

In addition to rate-independent elastoplasticity, ECI can also be applied to rate-dependent viscoelasticity. Here we model viscoelasticity based on a decomposition of the deformation gradient, which is independent of the elastoplastic decompostion. At each time step, the deformation gradient $\boldsymbol{F}$ can be decomposed into two different ways (Figure 3.10)

$$\boldsymbol{F} = \boldsymbol{F}^E \boldsymbol{F}^P = \boldsymbol{F}^N \boldsymbol{F}^V, \tag{3.39}$$

where $\boldsymbol{F}^N$ is the non-equilibrated elastic deformation gradient, and $\boldsymbol{F}^V$ is the viscous deformation gradient. $\boldsymbol{F}^N$ and $\boldsymbol{F}^E$ provide elastic responses additively. The evolution of $\boldsymbol{F}^N$ follows a similar principle as $\boldsymbol{F}^E$, which is characterized by a return mapping-like projection in the discrete setting. We follow the formulation of Fang et al. (2019):

$$\mathcal{Z}(\boldsymbol{F}^{N,tr}) = \boldsymbol{U}(A(\boldsymbol{\epsilon} - B\,\mathrm{tr}(\boldsymbol{\epsilon})\boldsymbol{I}))\boldsymbol{V}^\top, \tag{3.40}$$

where $A = \frac{1}{1+\Delta t\alpha}, B = \frac{\Delta t\beta}{1+\Delta t(\alpha+d\beta)}, \alpha = \frac{2\mu_N}{v_d}, \beta = \frac{2(2\mu_N+\lambda_N d)}{9v_v} - \frac{2\mu_N}{v_d d}$, and $\boldsymbol{F}^{N,tr}$ is the elastic predictor assuming no viscosity:

$$\boldsymbol{F}_p^{N,tr} = (\mathbf{I} + \Delta t^n \nabla \hat{\mathbf{v}}_p^{n+1})\boldsymbol{F}_p^{N,n}. \tag{3.41}$$

Here v_v and v_d are viscosity parameters , and μ_N and λ_N are independent Lamé parameters for viscoelasticity to the ones for elastoplasticity. For simplicity, we use $v_v = v_d = 2\mu_N v$ for some v.

Although the return mapping for viscoelasticity is totally different from the one for elastoplasticity, the vector field

$$\frac{\partial \Psi^N}{\partial \boldsymbol{F}^N}(\mathcal{Z}(\boldsymbol{F}))\mathcal{Z}(\boldsymbol{F})^\top \boldsymbol{F}^{-\top}$$

turns out to be integrable if Ψ^N is from the St. Venant-Kirchhoff constitutive model, and the augmented ECI energy for this vector field is

$$\Psi^{\text{Visco}}(\boldsymbol{\Sigma}) = \hat{\mu}\,\text{tr}((\log \boldsymbol{\Sigma})^2) + \frac{\hat{\lambda}}{2}(\text{tr}(\log \boldsymbol{\Sigma}))^2, \tag{3.42}$$

where $\hat{\mu} = A\mu_N$ and $\hat{\lambda} = A\lambda_N - AB(2\mu_N + d\lambda_N)$.

Without plasticity, $\boldsymbol{F}^P \equiv \boldsymbol{I}$, and then the strain energy density for a viscoelastic material is simply

$$\Psi(\boldsymbol{F}) = \Psi^E(\boldsymbol{F}) + \Psi^{\text{Visco}}(\boldsymbol{F}). \tag{3.43}$$

With MPM discretization, each particle p independently tracks the evolutions of $\boldsymbol{F}_p^N$ and $\boldsymbol{F}_p^E$ and independently updates them accordingly at the end of each time step.

3.1.5 Spatial-Temporal Integration

In this section, we present the detailed pipeline of ECI applied to MPM. The algorithm stages from t^n to t^{n+1} are listed as follows:

1. **Particles-to-grid transfer.** Grid mass m_i^n and velocity $\mathbf{v}_i^n$ are transferred from particle mass m_p, velocity $\mathbf{v}_p^n$, and angular velocity information $\mathbf{C}_p^n$ with APIC (Jiang et al., 2015b).

2. **Optimize new grid velocity.** A series of optimization problems in the form of Equation (3.7) using ECI augmented energies are solved until the fixed-point iteration converges or the maximal number of iterations is reached. See Section 3.1.5.1.

3. **Grid-to-particles transfer.** The grid velocity $\hat{\mathbf{v}}_i^{n+1}$ from the time integration are transferred back to particles to update particle velocity $\mathbf{v}_p^{n+1}$ and angular velocity information $\boldsymbol{C}_p^{n+1}$.

4. **Particle strain update.** The elastic strain $\boldsymbol{F}^E$ or $\boldsymbol{F}^N$ are updated according to return mappings.

5. **Particle advection.** Particles are advected via particle velocity: $\boldsymbol{x}_p^{n+1} = \boldsymbol{x}_p^n + \mathbf{v}_p^{n+1}\Delta t$.

We only elaborate on the second stage in the following section. The other stages are the same as the standard explicit MPM simulation pipeline (Jiang et al., 2016).

3.1.5.1 Iterative Stress Optimization Time Integration

To make the internal force of implicit plasticity integrable, the yield stress is viewed as constant in the force formulation, i.e., each particle sees a local cylinder-like yield surface with a different yield stress. Multiple optimizations with updated yield stresses are needed to make the final computed stresses consistent with the true yield surface. Each optimization problem is solved robustly using the projected Newton method with backtracking line search (Wang et al., 2020a), where the Hessian matrix is projected to a nearby positive definite form (Teran et al., 2005). See Algorithm 1 for the pseudo-code.

The update procedure of $\boldsymbol{\tau}_Y^{tr}$ can be viewed as a fixed point iteration:

$$\boldsymbol{\tau}_Y^{tr,j+1} = \boldsymbol{\Gamma}_Y^{tr}(\boldsymbol{F}^{E,tr}(\Delta\mathbf{v}(\boldsymbol{\tau}_Y^{tr,j}))). \tag{3.44}$$

Algorithm 1 Iterative Stress Optimization Time Integration

procedure MPMTimeIntegration($\Delta\mathbf{v}^{\text{init}}, \boldsymbol{M}^n, \mathbf{v}^n, \Delta t, \epsilon$) ▷ $\boldsymbol{M}^n$, $\mathbf{v}^n$ are stacked grid mass and velocity, $\Delta\mathbf{v}^{\text{init}}$ is the initial guess
  $\Delta\mathbf{v} = \Delta\mathbf{v}^{\text{init}}$
  do ▷ Iterative Stress Iteration
    for each particle p
      Evaluate trial hardening state q_p^{tr} ▷ Equation (3.38)
      Evaluate trial yield stress $(\tau_Y^{tr})_p$ ▷ Equation (3.35) Equation (3.38)
    end
    do ▷ Solve Equation (3.7)
      $\mathbf{r} \leftarrow -\nabla E(\Delta\mathbf{v})$ ▷ E as in Equation (3.7)
      $\delta\Delta\mathbf{v} \leftarrow \text{InexctMINRES}(\text{ProjectPD}(\nabla^2 E(\Delta\mathbf{v})), \mathbf{r})$ ▷ Section 3.1.5.3
      $\alpha \leftarrow \text{InversionFreeFilter}(\delta\Delta\mathbf{v})$ ▷ Section 3.1.5.4
      $E_{init} \leftarrow E(\Delta\mathbf{v})$
      while $E(\Delta\mathbf{v} + \alpha\delta\Delta\mathbf{v}) > E_{init}$ ▷ Line search
        $\alpha \leftarrow \frac{\alpha}{2}$
      end
      $\Delta\mathbf{v} \leftarrow \Delta\mathbf{v} + \alpha\delta\Delta\mathbf{v}$
      $\hat{\boldsymbol{r}} = \text{Residual}(\mathbf{r})$ ▷ Section 3.1.5.5
    while $\|\hat{\boldsymbol{r}}\|_\infty > \epsilon$
  while yield stress not converged
  for each particle p ▷ Advance hardening state
    $q^{n+1} = q_p^{tr}$
  end
end procedure

Here j is the index of stress iteration, $\Delta\mathbf{v}(\boldsymbol{\tau}_Y^{tr,j})$ is the equilibrated grid velocity field returned by a single optimization based on the yield stress vector $\boldsymbol{\tau}_Y^{tr,j}$, and the bold symbol represents the stacked stress vector from all particles or all grid nodes. Since the Jacobian of this iteration has a scalar Δt^2 (see the supplemental document for details), the convergence of this fixed-point iteration is guaranteed if Δt and the residual for the equilibrium are both small enough. In practice, we find that even with large time steps, only several fixed-point iterations are required to produce visually high-quality results.

3.1.5.2 Boundary Conditions

The boundary conditions in our simulations are all from rigid collision objects. At the beginning of each time step, we detect the set of grid nodes colliding with the collision objects

and directly enforce the velocity continuity condition across the collision interface. In each Newton iteration, the linear solver is projected so that the solved search direction remains tangent to the constraint manifold.

3.1.5.3 Inexact Newton-Krylov Methods

Following Wang et al. (2020a), we use an inexact Newton-Krylov method. The tolerance for the linear systems is set relatively large in an adaptive way. Although more Newton iterations are needed, the reduced linear solve cost can still improve the world-clock timing of Newton convergence. Specifically, we use matrix-free Minimal Residual Method (MINRES) to solve the linear systems and the relative tolerance of each MINRES solve is set to $\min(0.5, \max(0.1, \sqrt{\mathbf{r}^\top \boldsymbol{P}\mathbf{r}}))$, where $\mathbf{r}$ is the right-hand side vector and $\boldsymbol{P}$ is the preconditioning matrix.

3.1.5.4 Inversion-free Line Search

The Hencky strain requires that the deformation gradient is not inverted, i.e., $\det(\boldsymbol{F}^{E,tr}) > 0$. Following (Smith and Schaefer, 2015; Li et al., 2020, 2021e), before the line search, we first compute a large admissible step size α for the search direction such that the energy is well-defined for any step size $t \in [0, \alpha]$, and then the backtracking procedure starts with the filtered step size α.

3.1.5.5 Stopping Criteria

To terminate the Newton iterations early while ensuring visually high-quality simulation results, we normalize the grid residual vector $\mathbf{r}$, the gradient of the system energy, by the grid mass vector. This gives a residual in the unit of velocity (m/s), which is associated with a physical meaning. However, due to numerical rounding errors, small-mass nodes sometimes can have large residuals but contribute little to the particle advection. Therefore, we use grid-to-particle transfer to transfer the grid residual vector onto the particles to get the final residual vector. All our examples are running with tolerance $10^{-2}m/s$ based on the infinity

norm of the velocity-unit residual vector on particles.

3.1.5.6 Timestep Size Restriction

The time step size of MPM is bounded by the advection CFL condition (Gast et al., 2015). For those without stress-iterations, no further restrictions are required for our optimization integrator. For those with stress-iterations, theoretically, there is indeed a timestep size restriction for the stress iteration to fully converge, but we have not observed non-converging cases.

3.1.6 Discretization with FEM

ECI is independent of spatial discretization choices. Hence it can also be conveniently applied in Finite Element Methods (FEM).

In FEM, the conservation-of-momentum equation (Equation (3.3)) is directly discretized and solved in the material space. For FEM with linear tetrahedral elements, the discretized nodal internal force is

$$\boldsymbol{f}_i = -\sum_e V_e^0 \boldsymbol{P}_e \nabla N_{ie}, \tag{3.45}$$

where e indices all tetrahedral elements, V_e^0 is the rest volume of element e, and ∇N_{ie} is the gradient of the shape function on node i evaluated at the barycenter of element e (Irving et al., 2006).

Considering implicit plasticity, the internal force can be written as (see the supplemental document for details)

$$\boldsymbol{f}_i^{n+1} = -\sum_e V_e^0 \frac{\partial \Psi^E}{\partial \boldsymbol{F}^E}(\mathcal{Z}(\boldsymbol{F}_e^{E,tr}))\mathcal{Z}(\boldsymbol{F}_e^{E,tr})^\top \boldsymbol{F}_e^{E,tr^{-\top}} \boldsymbol{F}_e^{P,n^{-\top}} \nabla N_{ie}. \tag{3.46}$$

The integrability of the vector field $\frac{\partial \Psi^E}{\partial \boldsymbol{F}^E}(\mathcal{Z}(\boldsymbol{F}^{E,tr}))\mathcal{Z}(\boldsymbol{F}^{E,tr})^\top \boldsymbol{F}^{E,tr^{-\top}}$ leads us to the integrable

Table 3.1: Simulation Statistics.

Scene	Figure	Model	Δt (s)	Δx (m)	Geometry	Elasticity	Plasticity/Viscosity	s/Step
Sand castle	Figure 3.15	Drucker-Prager	0.0004	0.007	2.26M particles	$E = 5 \times 10^6$	$\phi_f = 30°, c = 0.0025$	18
Snow ball	Figure 3.17, Figure 3.18	Drucker-Prager	0.001	0.01	1.00M particles	$E = 10^6$	$\phi_f = 30°, c = 0.0025$	10
Noddle	Figure 3.1	Von-Mises	0.001	0.01	2.07M particles	$E = 10^6$	$\tau_Y = 7.7 \times 10^2, \xi = 0$	31
Hydraulic test (Can)	Figure 3.1, Figure 3.19a, Figure 3.19b	Von-Mises	0.01	/	156K elements	$E = 7 \times 10^9$	$\tau_Y = 3 \times 10^7, \xi = 0.5$	5.6
Hydraulic test (Cylinder)	Figure 3.20a	Von-Mises	0.01	/	299K elements	$E = 7 \times 10^9$	$\tau_Y = 3 \times 10^7, \xi = 0.1$	15
Hydraulic test (Square)	Figure 3.20b	Von-Mises	0.01	/	230K elements	$E = 7 \times 10^9$	$\tau_Y = 3 \times 10^7, \xi = 0.1$	8.1
Armadillo	Figure 3.22	Von-Mises	0.01	/	121K elements	$E = 10^6$	$\tau_Y = 10^5, \xi = 0.5$	99
Car crash	Figure 3.21	Von-Mises	0.005	/	152K elements	$E = 2 \times 10^9$	$\tau_Y = 2.5 \times 10^6, \xi = 0.1$	51
Memory foam	Figure 3.24	Viscoelasticity	0.01	/	212K elements	$E = 10^3$	$E_N = 2 \times 10^5, v = 0.01$	17

internal force from the augmented elastoplastic energy density Ψ:

$$\boldsymbol{f}_i^{n+1} = -\frac{\partial}{\partial \boldsymbol{x}_i}(\sum_e \Psi(\boldsymbol{F}_e^{E,tr})V_e^0), \tag{3.47}$$

where x_i is the world space coordinate of node i.

At the end of each time step, we need to track and update $\boldsymbol{F}^P$ on each element with

$$\mathcal{Z}(\boldsymbol{F}^{E,tr})\boldsymbol{F}^{P,n+1} = \boldsymbol{F}^{E,tr}\boldsymbol{F}^{P,n}. \tag{3.48}$$

ECI combined with Incremental Potential Contact (IPC) (Li et al., 2020) allows us to simulate various scenarios where both accurate frictional contacts and inelastic responses are essential.

Timestep Size Restriction Similar to MPM, there is also a timestep size restriction for the stress iterations to fully converge in FEM. Other than that, no further restrictions are needed. However, there is certainly a tradeoff between the number of timesteps and the accuracy and overall efficiency of the simulation (Li et al., 2020), which holds for all time discretized numerical schemes.

3.1.7 Evaluation

We demonstrate the versatility of ECI with both MPM and FEM simulations. Among these examples, the ones that do not contain topological changes are simulated with FEM, and the frictional contact is modeled with IPC (Li et al., 2020). For our MPM simulations, we use a

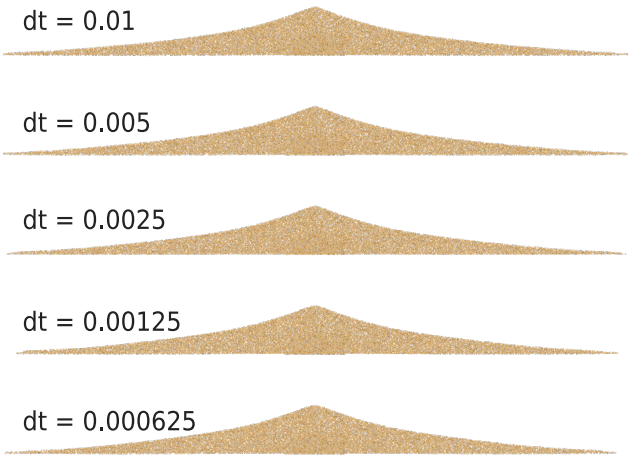


Figure 3.11: 2D sand column collapse experiment with consecutively halved time steps.

Δt	# Stress Iter. (Avg. / Max)	# Newton Iter. (Avg. / Max)	# Line search (Avg. / Max)
0.01	8.8 / 13	112.3 / 186	202.0 / 476
0.005	6.8 / 9	45.2 / 75	48.8 / 201
0.0025	5.1 / 7	18.4 / 34	9.6 / 60
0.00125	3.7 / 6	9.2 / 14	1.3 / 9
0.000625	2.6 / 4	4.7 / 8	0.0 / 0

Table 3.2: Simulation statistics of 2D sand column collapse experiment

CFL number of 0.6 (Gast et al., 2015). The world-clock timing and the simulation setup are reported in Table Table 3.1. The statistics are based on Intel Core i9-10920X 3.5-GHz CPU with 12 cores.

3.1.7.1 Unit Tests

Convergence of Stress Iteration. We test the convergence of the stress iteration on a 2D sand column collapse experiment. We use a direct solver to solve linear systems in the optimization time integrator, to avoid complicating the experiments with possibly inexact Krylov solves. The convergence criteria of the stress iteration is $\|(\tau_Y^{j+1} - \tau_Y^j)\|_2 < 10^{-9}(2\mu\sqrt{N})$, where N is the number of particles, and the Newton tolerance is 10^{-5}. Note that these tolerances are much tighter than needed so that we can verify that our method can converge with high accuracy. We consecutively halve the time step size from $\Delta t = 0.01s$. All these tests successfully converge with the given convergence criteria and have consistent results

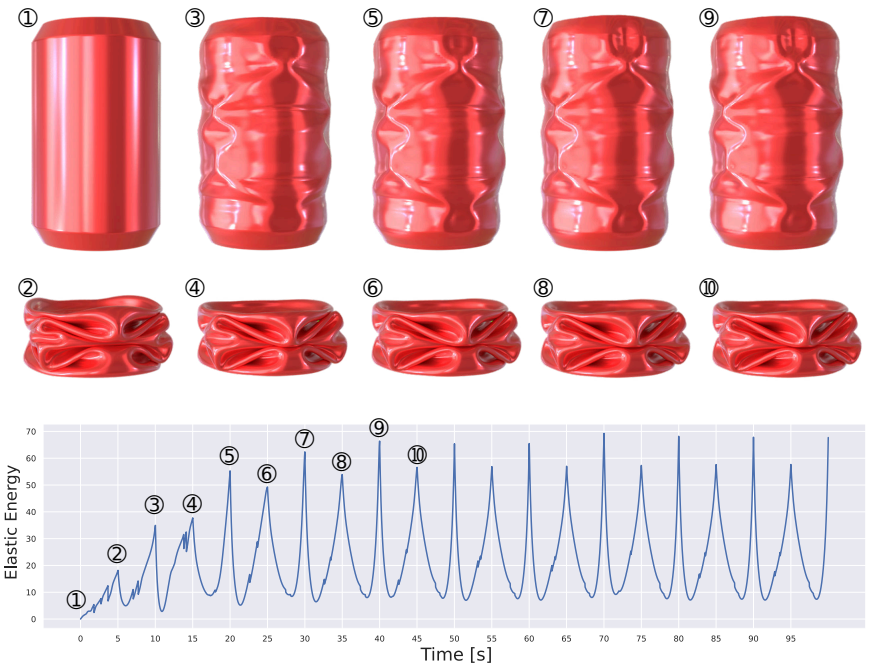


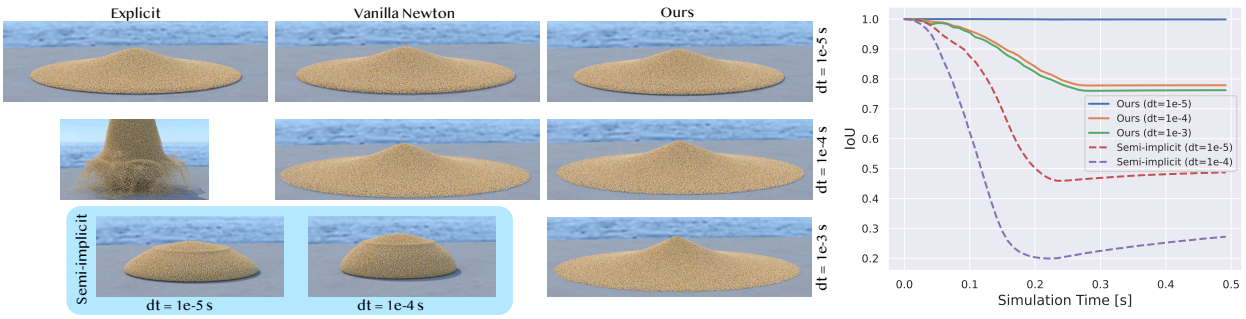
Figure 3.12: With lower and upper bounds, the stored elastic energy in the soda can changes periodically over time during the compressing–stretching cycles, which demonstrates the long-time stability of our simulation.

(see Figure 3.11). The iteration statistics are listed in Table 3.2, which shows that as the time step size decreases, the required number of stress iterations, Newton iterations, and line searches all decrease as expected.

Long-Time Stability. To test the long-time stability of our method, we simulate a soda can being periodically compressed and stretched 10 cycles with $\Delta t = 10^{-2}s$ (Figure 3.12). The Young's modulus of the soda can is 7 GPa. The stored elastic energy over time is always bounded and it oscillates along with the compressing–stretching cycles, demonstrating the strong long-time stability property of our method.

3.1.7.2 Comparisons to Explicit and (Semi-)Implicit Plasticity

We compare our variational method with both explicit and implicit methods proposed in Klár et al. (2016) on a 3D *sand column collapse* experiment. The time step size Δt for these two methods both need fine-tuning to avoid numerical explosion. For explicit integration, the



(a) (b)

Figure 3.13: **Sand Column Collapse. (a)** The explicit method explodes with $\Delta t = 10^{-4}s$. The implicit method (Klár et al., 2016) with vanilla Newton fails at a time step where the scene almost becomes static $\Delta t = 10^{-4}s$ (the frame right before the failure is rendered here). Our method works with all three time step sizes $\Delta t = 10^{-5}s, 10^{-4}s, 10^{-3}s$, and produces consistent results. The semi-implicit method produces artificial elastic behaviors even with a small time step size. **(b)** With the explicit method as the ground truth, our method has a smaller error (larger IoU score) than the semi-implicit method.

time step is bounded by the sound-speed CFL (Sun et al., 2020b), which is small in general, especially for stiff materials and at high resolution. With Klár et al. (2016)'s implicit method based on the non-integrable implicit force (Equation (3.12), with asymmetric force Jacobian), the convergence of time integration can only be reached if the initial guess is sufficiently close to the local optimum. Furthermore, the search performed by the Newton-Raphson iterations (we refer it as the vanilla Newton method) can result in deformation gradients with non-positive determinants that cause simulation failure.

We experiment under three different time step sizes $\Delta t = 10^{-3}s$, $10^{-4}s$ and $10^{-5}s$. Explicit MPM can run with $\Delta t = 10^{-5}s$, but it explodes with $\Delta t = 10^{-4}s$ (Figure 3.13a top left). The vanilla Newton method can run with $\Delta t = 10^{-5}s$, but fails at a step when the simulation almost becomes static with $\Delta t = 10^{-4}s$ and at the first step with $\Delta t = 10^{-3}$ (Figure 3.13a top middle). Our method, on the other hand, works well with all these three time step sizes and produces consistent results (Figure 3.13a right).

A common heuristic treatment is to directly replace the grid update step in the explicit MPM simulation with implicit time integration without plasticity and only conduct return mappings at the end of the time steps. We refer to this elasticity-plasticity-decoupled scheme as the semi-implicit method in this paper. Although its stability and convergence can be

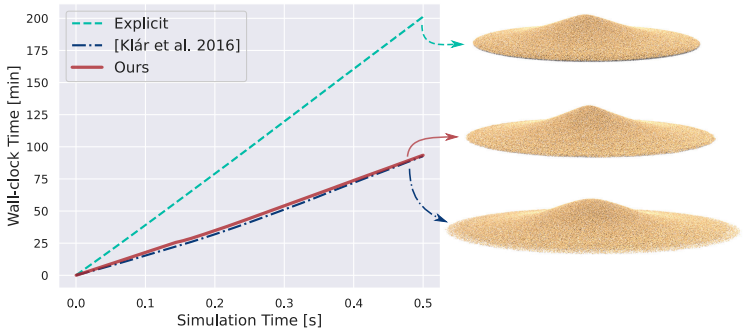


Figure 3.14: ECI achieves 2× speedup compared to the explicit method in a sand column collapse experiment and is with comparable speed to the implicit method in Klár et al. (2016).

guaranteed by the optimization time integration, the semi-implicit method can lead to severe artifacts as shown in Figure 3.13a bottom left, where the forces provided by the stresses outside the yield surface make the continuum behave more like a purely elastic body. This is due to the ignorance of the plasticity by the implicit solve, which in turn overestimates the material's resistance to tensile deformation. Our method, on the other hand, fully resolves plasticity in the implicit solve and does not suffer from any such artifacts. We use the explicit method as the ground truth to quantitatively measure errors. Figure 3.13b shows that our method has a smaller error than the semi-implicit method, where we compute the Intersection over Union (IoU) metrics between MPM grid mass distributions (computed as the ratio of the number of common grid nodes to the number of union grid nodes).

In practice, we can limit the number of Newton iterations and Krylov iterations. On a *sand column collapse* experiment with the same physical parameters and initial setup as above, our method with $\Delta t = 2 \times 10^{-5} s$ achieves 2× speedup compared to the explicit method with $\Delta t = 10^{-5} s$, as shown in Figure 3.14. With 2 stress iterations per time step, 1 Newton iteration per stress iteration, and 5 MINRES iterations per Newton iteration, our method can still generate physically plausible results. To make it a fair comparison, the maximal numbers of Newton iterations and GMRES iterations are set to 2 and 10 respectively for Klár's implicit method with the same $\Delta t = 2 \times 10^{-5} s$ as ours. The simulation using Klár's method does not go unstable in this setting, and its computational cost is similar to ours, as expected.

Figure 3.15: **Snow castle.** With our variational inelasticity model, the castle can be smashed into pieces after hitting by the fish, while with the semi-implicit method the castle behaves like an elastic body, holding the fish in an unrealistic way.

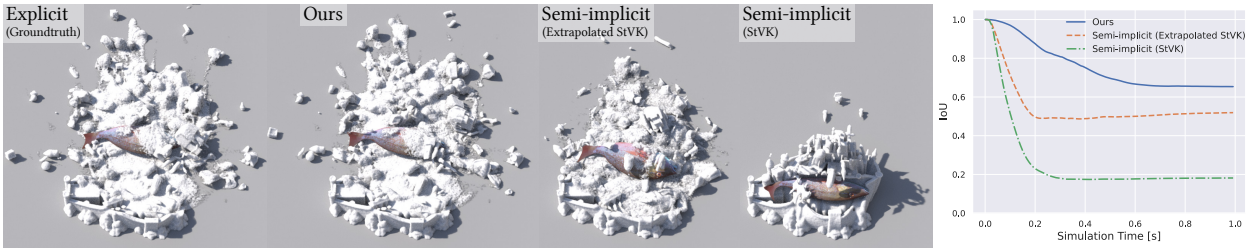


Figure 3.16: Our method is more accurate visually and quantitatively than the semi-implicit methods with/without the extrapolated StVK constitutive model.

3.1.7.3 Druker-Prager Plasticity with Cohesion

Snow Castle To further demonstrate the artifacts caused by fully decoupling elasticity and plasticity, we simulate a snow castle hit by a high-speed elastic fish. The snow is modeled with wet soil by Druker-Prager plasticity with cohesion. With our variational model, the fish smashes the snow castle into pieces completely. However, with the semi-implicit method, the castle behaves like an elastic body and ends up holding the fish in an unrealistic way.

Our extrapolated StVK constitutive model combined with the volume-preserving return mapping plays a vital role in generating fractures in this example. Intuitively, our scheme mimics the cohesion behavior better because it allows particles to be compressed a little before exerting resisting force. Under the same time step size ($\Delta t = 5 \times 10^{-5}$), we use the result from the explicit method with the extrapolated StVK model as the ground truth to compare the accuracy between our method and semi-implicit methods (with/without the extrapolated StVK model). The visual and quantitative comparisons in Figure 3.16 both show that our method is more accurate.

Figure 3.17: **Snow Ball.** A free-falling snow ball hits on a static dragon and smashes into pieces.

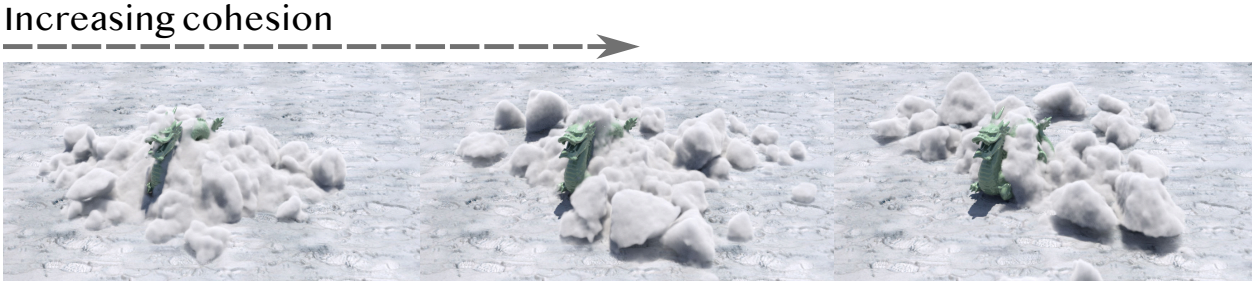


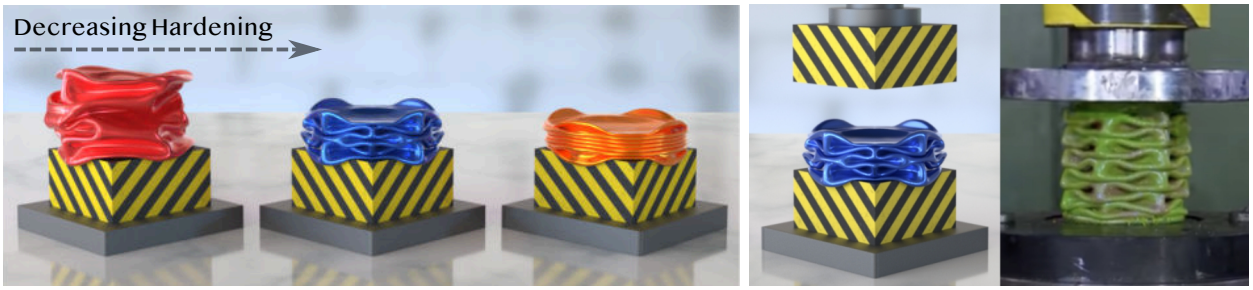
Figure 3.18: Larger cohesion strength increases the chunkiness of the snow. From left to right, $c = 0.00125, 0.0025, 0.005$.

Snow Ball We use the Druker-Prager plasticity model with cohesion to simulate a snow ball hitting a static dragon (Figure 3.17). We also simulate with different cohesion strengths to show the controllability of our method on simulating different levels of chunkiness (Figure 3.18).

3.1.7.4 Von-Mises Plasticity (with Hardening)

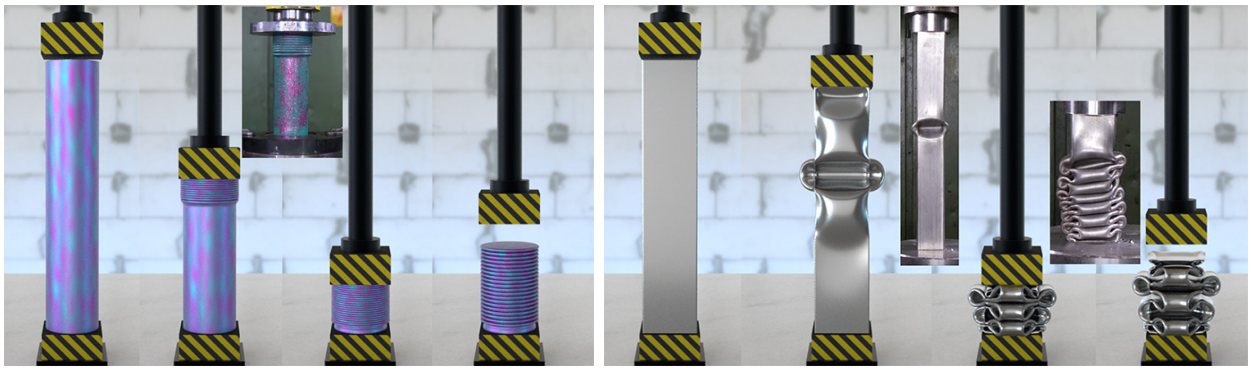
Play-Doh Noodle. MPM can automatically handle topology changes. By leveraging this feature, we simulate a Play-Doh modeled by the von-Mises plasticity pressed through a cylindrical noodle mold (Figure 3.1 (top row)).

Hydraulic Tests on Metals. Hardening is widely observed in metals. We simulate hydraulic tests on soda cans with different hardening coefficients and compare with the simulation without hardening (Figure 3.1 bottom row). The hardening mechanism makes plastic deformations harder to happen as the yield surface expands. This lets the object restore its original rest shape partially when all boundary conditions are released. As shown



(a) (b)

Figure 3.19: **(a)** Different hardening coefficients lead to varying restorations towards the rest shape and generate different crushing patterns. From left to right, the hardening coefficient $\xi = 0.5, 0.3, 0$. **(b)** One of our hydraulic test simulations on metal cans generate a crushing pattern well matching that in a real video footage (Youtube, 2021).



(a) (b)

Figure 3.20: **Hydraulic Tests on Metal Pipes.** The crushing pattern matches the results of real-world experiments. (Youtube, 2018, 2021).

in the last frame when the upper press withdraws (Figure 3.19a), the red can with the largest hardening coefficient restores the most, and the orange can with no hardening almost does not restore at all. Furthermore, different hardening coefficients generate different crushing patterns. As shown in Figure 3.19b, the deformation patterns in one of our compressed can match that from a real experiment. Modeling hardening also allows us to successfully capture the snap-through instability of metal, which can be observed in real experiments (see our video demonstration). When we swap in long steel pipes for the hydraulic tests (one cylindrical, one square), the crushing patterns also match real experiments well (Figure 3.20).

Figure 3.21: **Car collision.** The red car hits the yellow car against the wall by 56 mph, creating a big dent on the right side of the yellow car.

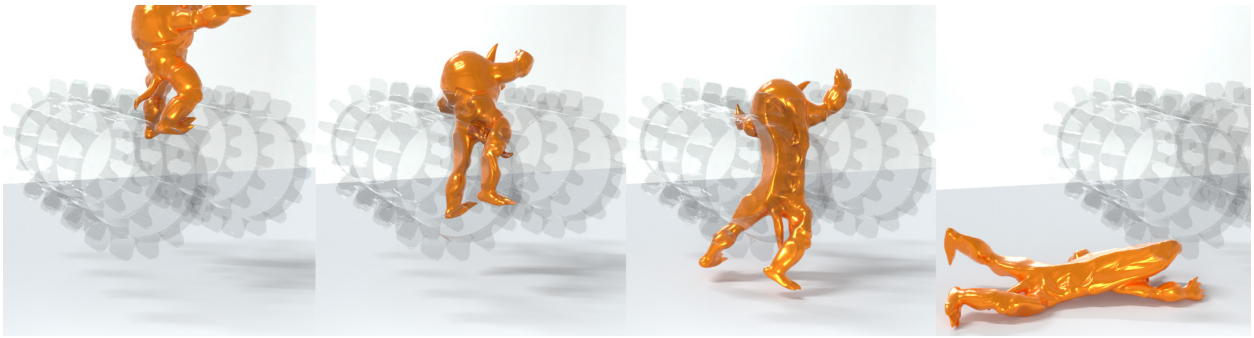


Figure 3.22: **Squeeze armadillo.** Rolling a plastic armadillo through a gear left the gear teeth permanently distorts the armadillo body.

Car Crash and Crushed Armadillo. To further demonstrate the hardening behaviors of metals, we simulate a high-speed car crashing into another stationary car (Figure 3.21) and an armadillo rolling through a metal crusher driven by frictions (Figure 3.22). Both examples show realistic denting effects with sufficient restoration towards the rest shape enabled by hardening.

Comparison to Semi-Implicit Plasticity. We simulate a stiff elastic ball hitting a wall modeled by the von-Mises plasticity to compare our method with the semi-implicit method. As shown in Figure 3.23, the permanent deformations of the wall clearly show that the semi-implicit plasticity overestimates the material's resistance. We use the explicit method as the ground truth to compare the position error of the wall (computed as the average squared norm of vertex position differences), which shows that our method is more accurate than the semi-implicit method.

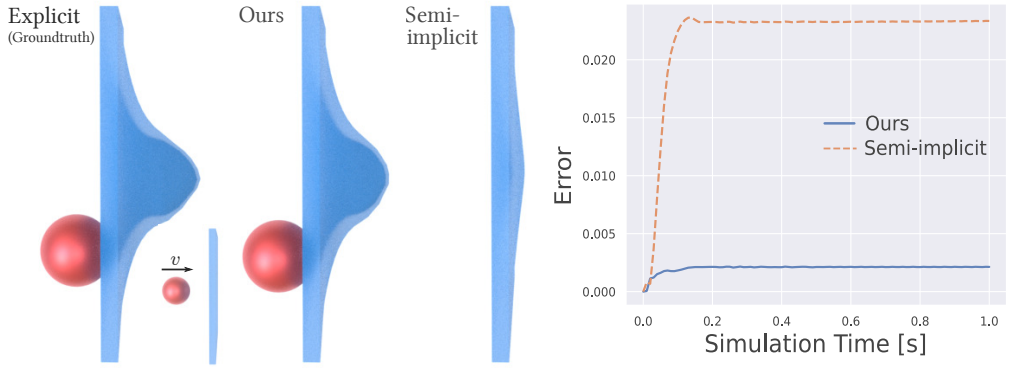


Figure 3.23: The semi-implicit von-Mises plasticity (right) overestimates the resistance response and results in a large error compared to the ground truth (left). Ours (middle) is much more accurate.

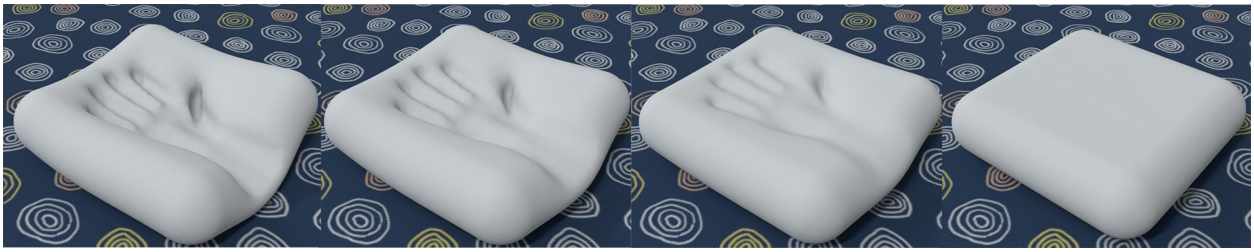


Figure 3.24: **Memory Foam.** A hand presses down a memory foam pillow for a while to leave a hand print, and then disappears suddenly. From left to right, the hand print slowly disappears as the deformed memory foam gradually restores its initial rest shape.

3.1.7.5 Viscoelasticity

Memory foam is a typical material demonstrating the viscoelastic behaviors in the real world. We simulate a pillow made by memory foam pressed down by a hand for a while, and then we lift the hand suddenly. We successfully capture the intricate process where the pillow slowly recovers its rest shape, completely removing the imprint of the hand (Figure 3.24).

3.1.8 Discussion

In summary, we developed ECI, a new formulation that augments hyperelastic energy density functions to enable variational forms for a wide range of elastoplastic and viscoelastic materials. Our algorithm enables the fully implicit simulation of inelasticity in recently advanced optimization-based time integrators, embracing advantages of long-time stability,

global convergence, large time step sizes, and high accuracy.

Our method is most naturally "plug and play" when applied to J2 von Mises materials and finite strain viscoelastic materials. However, when extended to pressure-dependent plasticity or strain hardening mechanisms, additional iterations on the stress are necessary to achieve final convergence. In our examples, usually 1-2 stress iterations are sufficient to generate convergent or visually high-quality results. It is promising future work to devise theoretical and algorithmic improvements to guarantee and accelerate the convergence, particularly for accuracy-demanding applications.

The integrability of the implicit elastoplastic force depends on both the elastic model and the plastic model. For instance, although the combination of St. Venant-Kirchhoff elasticity with von-Mises plasticity adopted by ECI leads to a symmetric force Jacobian, neo-Hookean elasticity with von-Mises plasticity does not. It is an interesting future work to explore integrable approximations to other combinations.

ECI assumes the full-dimensional volumetric deformation gradient. Accordingly, our metal cans and pipes are all simulated with thin single-layer linear tetrahedral elements, which could potentially suffer from shear locking. It would be interesting to extend ECI to codimensional geometries like shells and rods (Narain et al., 2013).

Finally, our augmentation to the strain energy density function changes the conditioning of the global stiffness matrix. It is interesting future work to study its effect on the linear solve, and strategies to precondition the ECI-augmented system.

3.2 PlasticityNet: Learning to Simulate Metal, Sand, and Snow for Optimization Time Integration

3.2.1 Introduction

Combining machine learning with physical simulations has recently attracted a lot of attention. A vast amount of existing research adopts an end-to-end approach, where the specific underlying computational physics system is treated as a black box (Sanchez-Gonzalez et al.,

2020; Pfaff et al., 2020). Harnessing the power of neural networks, this research has been successfully applied in computer animation (Eckert et al., 2019), multibody systems (Battaglia et al., 2016; Chang et al., 2016; Yang et al., 2020; Gan et al., 2020), human musculature simulation (Jin et al., 2022), computational fluid dynamics (Belbute-Peres et al., 2020; Garnier et al., 2021), and non-linear continuum mechanics (Bock et al., 2019). An alternative direction is represented by physics-informed neural networks (PINN) (Raissi et al., 2019; Karniadakis et al., 2021), where in its original form, the residual of a partial differential equation is directly used as the loss function so that the network training is a physics-aware learning process. PINN becomes powerful when the design space of the input to the network can be parameterized, which accelerates both the roll-out and the inverse optimization process (Sun et al., 2020a). Another noteworthy category is learning the physical modeling where the machine can either help increase the model resolution in a coarser grid (Kochkov et al., 2021), inject nonlinearity to a linear model (Luo et al., 2018), or apply a learnable model reduction to reduce the system degrees-of-freedom (DOF) for acceleration (Shen et al., 2022, 2021).

Despite its great success, training a neural network to replace a traditional simulator is not always the preferred choice. This is partially due to the challenges in the trained model's generality and portability. For example, a trained model on a particle-based deformable body solver (such as the Material Point Method (MPM (Jiang et al., 2016)) cannot be directly applied to the mesh-based Finite Element Method (FEM) (Sifakis and Barbic, 2012), while in traditional continuum mechanics, the constitutive model that describes the relationship between force and deformation is an independent module from the underlying geometric description or simulation scheme. Indeed, by simply switching the constitutive model and applying minor changes to the existing and general simulation pipeline, a wide range of materials can be simulated in the same framework, ranging from sand (Prager, 1955; Drucker, 1950; Klár et al., 2016; Tampubolon et al., 2017) and metal (Mises, 1913a), to snow (Gaume et al., 2018; Wolper et al., 2019; Li et al., 2021d) and glacier (Wolper et al., 2021).

Many elastic materials, including those represented by mass-spring systems (Bargteil et al., 2020) and common *hyperelastic* solids (Stomakhin et al., 2012), are usually governed by analytical elastic potential energy functions in terms of the deformation. These models are

well fitted to experiments and proven to be simple, accurate, and predictive. Although most of these energy functions are highly nonlinear and non-convex, reformulating the dynamic simulation process as a numerical optimization problem and solving it using projected Newton and line search can guarantee global convergence to a solution (Li, 2020). Beyond hyperelasticity, plasticity is much more challenging. The mechanical response of plastic materials imposes extra difficulties in the implementation as it is path-dependant and non-smooth. One common handling of plasticity is the return mapping algorithm, which applies the effects of plastic deformation to the elastic forces. However, this leads to asymmetrical force derivatives, which eliminate the possibility of integrating the plasticity into the energy function in a single optimization and complicates the pipeline. In the recent work of Energetic Consistent Inelasticity (ECI) (Li et al., 2022f), the plasticity is analytically modeled as an energy functional, and the simulation can be formulated as an optimization problem just like simulating pure elastic materials. However, their analytical derivation only works for St.Venant-Kirchhoff (StVK) elasticity with the von-Mises plasticity.

In this work, we propose PlasticityNet, a neural network-based approach for learning an energy-based force that locally approximates elastic forces with plasticity models and is compatible with optimization time integrators. PlasticityNet framework supports any combinations of elastic models and plastic models and works with both MPM and FEM discretizations. With optimization time integrators, we demonstrate that our framework can simulate vast types of plasticities, such as metal, sand, and snow, with large time step sizes.

3.2.2 Related work

Classic Plasticity Models The classic plastic models utilized the geometry information of the plasticity and are available for many applications. In the computer graphics community, researchers have followed mechanical literature on the Drucker-Prager elastoplasticity model (Prager, 1955; Drucker, 1950), and developed particle-based simulations of dry (Klár et al., 2016) and wet (Tampubolon et al., 2017) sand. Extending a similar Cam-Clay plasticity model, snow avalanches (Gaume et al., 2018; Li et al., 2021d), glacier calving (Wolper et al.,

2021) and food fracturing (Wolper et al., 2019) are also captured with high visual plausibility as well as physical accuracy. For metals and dough-like materials, the von-Mises plasticity model (Mises, 1913a) is usually adopted, while (Wolper et al., 2020; Jiang et al., 2017a) presented its anisotropic extensions. Still, the implementation of these models in modern, optimization-based simulators is cumbersome due to the non-integrable forces. Recently, (Li et al., 2022f) proposed an elastoplastic energy of von-Mises plasticity under StVK elasticity for optimization time integrator, which can be viewed as a special-case analytical solution to our framework under the same combination of elasticity and plasticity. But our framework works for arbitrary combinations.

Data-Driven Plasticity Models The machine learning approach has been used to find new plastic models using large sets of measurements and parameters, outperforming many long-standing hand-crafted models. The macro-level constitutive relationship is learned from the results of the micro-level simulations (Mozaffar et al., 2019; Reimann et al., 2019). A similar approach is applied in (Vlassis et al., 2020) to learn anisotropic hyperelasticity, where additional geometrical information is included in the input. PINN can also be applied in plastic model finding from experimental measurements (As' ad et al., 2022; Vlassis and Sun, 2021; Koeppe et al., 2022), where the loss includes the stress and Hessian, to infer stress with more accuracy in the implicit simulators. However, there does not exist any prior work, to the best of the author's knowledge, that tried to find variational form for arbitrary plasticity model.

Optimization Time Integration The optimization time integrators have advantages in terms of stability under large deformations and large time step sizes. Many of the nonlinear systems of equations that arise from implicit simulation can be integrated to get equivalent optimization problems, which allow robust optimization techniques to be applied. The MPM simulator in this work is based on (Gast et al., 2015), which formulated the backward Euler time integration with hyperelastic materials as a minimization problem. (Li et al., 2019a) and (Wang et al., 2020a) also explored domain decompositions and hiearachical

preconditioners to improve robustness and efficiency. The FEM simulator in this work is based on Incremental Potential Contact (IPC) (Li et al., 2020), which proposed a variational form for frictional contacts. Their optimization-based frictional contact framework was also extended to codimensional objects (Li et al., 2021a), rigid bodies (Ferguson et al., 2021a; Lan et al., 2022b), articulated multibodies (Chen et al., 2022b), reduced elastic solids (Lan et al., 2021a), embedded interfaces (Zhao et al., 2022), and FEM-MPM coupled domains (Li et al., 2021c).

3.2.3 Background

3.2.3.1 Optimization Time Integration

In this section, we briefly introduce the optimization time integration for elastodynamics simulations with the Material Point Method (MPM) and the Finite Element Method (FEM). We refer the readers to (Li, 2020) and (Gast et al., 2015) for more details.

FEM discretizes the simulation domain as unstructured meshes (e.g., triangle meshes in 2D), while in MPM, a point cloud composed of material particles is used to discretize the domain. While FEM directly uses the mesh nodes as the simulation degrees-of-freedom (DOF), MPM transfers its particle state to a uniform background grid, whose nodes are used as the DOFs for the integration of forces (Jiang et al., 2016). Robust simulation of elastodynamics can be achieved via implicit time integration, which updates the nodal positions ($\mathbf{x}$) or velocities ($\mathbf{v}$) step by step based on the previous physical states. To step from t^n to $t^{n+1} = t^n + \Delta t$ with time step size Δt, with implicit Euler time integration rule, one needs to solve a nonlinear system of equations

$$\mathbf{M}(\mathbf{v}^{n+1} - (\mathbf{v}^n + \mathbf{g}\Delta t)) = \Delta t \mathbf{f}^{n+1}. \tag{3.49}$$

Here $\mathbf{v}$ is the velocity DOF formed by concatenating all nodal velocity vectors, similarly concatenated, $\mathbf{M}$ is the mass matrix, $\mathbf{g}$ is the gravitational acceleration vector, and $\mathbf{f}$ is the

internal force vector. Without plasticity, the internal force on a node i can be calculated as

$$\mathbf{f}_i^{n+1} = -\sum_q V_q^0 \mathbf{P}(\mathbf{F}_q^{n+1}) \nabla w_{iq}, \tag{3.50}$$

where q iterates the surrounding elements/particles of node i in FEM/MPM, V_q^0 is the initial volume of the element/particle, $\mathbf{F} = (\mathbf{I} + \Delta t \nabla \mathbf{v})\mathbf{F}^n$ (MPM) or $\mathbf{F} = \nabla \mathbf{x}^n + \Delta t \nabla \mathbf{v}$ (FEM) is the deformation gradient, which measures deformation from the undeformed state to the deformed state, and $\mathbf{P}$ is the first-Piola Kirchhoff stress, which describes the internal force per unit area within a material. ∇w_{iq} is the gradient of the weight function on node i evaluated on an element/particle center. The weight function is for transferring physical quantities between the elements/particles and the mesh/grid nodes. Unlike FEM, the last time step is used in MPM as the reference configuration, and so ∇w_{iq} is calculated as $\mathbf{F}_q^{n\top} \nabla w_{iq}^n$.

When there exists an energy density function Ψ such that $\mathbf{P}(\mathbf{F}) = \frac{\partial \Psi}{\partial \mathbf{F}}$, solving Equation (3.49) is equivalent to solving the following optimization problem

$$\mathbf{v}^{n+1} = \text{argmin}_{\mathbf{v}} \frac{1}{2} \|\mathbf{v} - (\mathbf{v}^n + \mathbf{g}\Delta t)\|_{\mathbf{M}}^2 + \sum_q V_q^0 \Psi(\mathbf{F}_q). \tag{3.51}$$

This formulation is more favored because with line search methods, convergence to a local minimum of Equation (3.51) can be guaranteed even when simulating challenging cases with stiff materials or large time step sizes. After solving for the velocity $\mathbf{v}^{n+1}$, FEM directly updates mesh nodal positions as $\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t \mathbf{v}^{n+1}$, while for MPM, the velocity on the grid node is interpolated to particle locations for particle advection. The background grid is reset at the beginning of each time step, which allows MPM to benefit from the conveniences of a regular grid and a mesh-free formulation at the cost of some accuracy loss due to the transfers between the grid and particles.

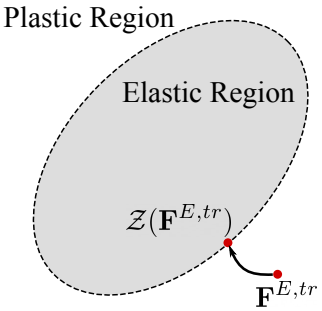


Figure 3.25: An illustration of a return mapping.

3.2.3.2 Return Mapping for Plasticity

With plasticity, objects can undergo both plastic and elastic deformations, and the deformation gradient at the current time step can be decomposed as

$$\mathbf{F}^{n+1} = \mathbf{F}^{E,n+1}\mathbf{F}^{P,n+1} \tag{3.52}$$

based on the finite strain theory. Here $\mathbf{F}^{P,n+1}$ encodes the permanent plastic deformation of the rest shape, and $\mathbf{F}^{E,n+1}$ is the elastic deformation that results in effective elastic forces. In theory, $\mathbf{F}^{E,n+1}$ is constrained within certain elastic regions. Computation-wise, an elastic predictor $\mathbf{F}^{E,tr} = \mathbf{F}^{n+1}(\mathbf{F}^{P,n})^{-1}$ can be computed first by assuming $\mathbf{F}^{P,n+1} = \mathbf{F}^{P,n}$. If $\mathbf{F}^{E,tr}$ is outside the elastic region, it will be projected back onto the boundary of the region to obtain $\mathbf{F}^{E,n+1} = \mathcal{Z}(\mathbf{F}^{E,tr})$ (Figure 3.25). This projection $\mathcal{Z}$ is called a *return mapping*. Within this framework, the implicit elastoplastic nodal force can be computed as (Li et al., 2022f)

$$\mathbf{f}_i^{n+1} = -\sum_q V_q^0 \boldsymbol{\tau}(\mathcal{Z}(\mathbf{F}_q^{E,tr}))\mathbf{F}_q^{E,tr^{-\top}}\mathbf{F}^{P,n^{-\top}}\nabla w_{iq} \tag{3.53}$$

where $\boldsymbol{\tau}(\mathbf{F}) = \mathbf{P}(\mathbf{F})\mathbf{F}^\top$ is the Kirchoff stress. The above forces are integrable only if $\boldsymbol{\tau}(\mathcal{Z}(\mathbf{F}))\mathbf{F}^{-\top}$ can be represented as the gradient of some energy function:

$$\frac{\partial \Psi}{\partial \mathbf{F}} = \boldsymbol{\tau}(\mathcal{Z}(\mathbf{F}))\mathbf{F}^{-\top}. \tag{3.54}$$

Most combinations of elastic constitutive models and plastic return mappings do not satisfy this integrability condition because the Jacobian field of the right-hand side is asymmetrical. Note that directly feeding $\mathcal{Z}(\mathbf{F})$ into an elastic potential does not form a potential energy for the elastoplastic forces defined in Equation (3.53). (Li et al., 2022f) only found one specific combination such that an elastoplastic potential energy exists. Thus, it remains challenging to simulate versatile plastic behaviors with optimization time integrators and achieve robust performance.

3.2.4 PlasticityNet

We propose PlasticityNet, a neural network-based elastoplastic model that finds a family of local potential energies whose negative gradients can approximate the elastoplastic forces within a small neighborhood so that plasticity can be conveniently simulated using optimization time integrators. The model architecture is illustrate in Figure 3.26. Specifically, instead of finding a global energy function $\Psi(\mathbf{F})$, we search for an energy $\Psi(\mathbf{F}, \mathbf{F}_0)$, parameterized by $\mathbf{F}_0$, such that

$$\frac{\partial \Psi}{\partial \mathbf{F}}(\mathbf{F}, \mathbf{F}_0)|_{\mathbf{F}=\mathbf{F}_0} = \boldsymbol{\tau}(\mathcal{Z}(\mathbf{F}_0))\mathbf{F}_0^{-\top}, \quad \text{and} \quad \frac{\partial \Psi}{\partial \mathbf{F}}(\mathbf{F}, \mathbf{F}_0) \approx \boldsymbol{\tau}(\mathcal{Z}(\mathbf{F}))\mathbf{F}^{-\top}. \tag{3.55}$$

To exactly enforce the first equality, we propose the following linear correction:

$$\Psi_\theta(\mathbf{F}, \mathbf{F}_0) = \mathcal{NN}_\theta(\mathbf{F}, \mathbf{F}_0) - (\nabla_{\mathbf{F}}\mathcal{NN}_\theta(\mathbf{F}_0, \mathbf{F}_0) - \boldsymbol{\tau}(\mathcal{Z}(\mathbf{F}_0))\mathbf{F}_0^{-\top}) \odot \mathbf{F}. \tag{3.56}$$

Here $\mathbf{A} \odot \mathbf{B} = A_{ij}B_{ij} = \mathrm{tr}(\mathbf{A}^\top\mathbf{B})$ is the matrix inner product. It can be verified that $\frac{\partial \Psi_\theta}{\partial \mathbf{F}}(\mathbf{F}, \mathbf{F}_0)|_{\mathbf{F}=\mathbf{F}_0} = \boldsymbol{\tau}(\mathcal{Z}(\mathbf{F}_0))\mathbf{F}_0^{-\top}$.

Then we only need to focus on the approximation part in Equation (3.55). We design the training loss function for our neural network as

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{F}_0}\mathbb{E}_{\mathbf{F}} \left\| \frac{\partial \Psi_\theta}{\partial \mathbf{F}}(\mathbf{F}, \mathbf{F}_0) - \boldsymbol{\tau}(\mathcal{Z}(\mathbf{F}))\mathbf{F}^{-\top} \right\|_F^2. \tag{3.57}$$

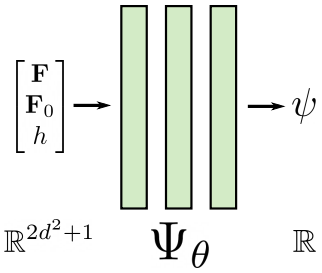


Figure 3.26: An overview of PlasticityNet. It is a map from $\mathbb{R}^{2d^2+1}$ to $\mathbb{R}$.

During training, $\mathbf{F}$ is only sampled near $\mathbf{F}_0$. Please refer to Section 3.2.5.1 for details.

3.2.4.1 Hardening of Plasticity

Hardening effects are widely observed in metals and snow. With hardening, the elastic region will expand by a certain amount whenever $\mathbf{F}^{E,tr}$ falls in the plastic region. To account for hardening, the return mapping $\mathcal{Z}(\mathbf{F}, h)$ and the energy $\Psi_\theta(\mathbf{F}, \mathbf{F}_0, h)$ will depend on an extra hardening state h, which controls the shape of the elastic region. This hardening state is a function of $\mathbf{F}$. However, to maintain integrability with respect to $\mathbf{F}$, we approximately update h based on $\mathbf{F}_0$, which is assumed to be close to $\mathbf{F}$.

3.2.4.2 Optimization Time Integration with PlasticityNet

Fixed-Point Iteration The gradient of our learned elastoplastic potential energy $\Psi_\theta(\mathbf{F}, \mathbf{F}_0)$ only approximates the effective stresses locally near $\mathbf{F}_0$. To approach the accurate solution of Equation (3.49) with elastoplastic forces, we apply a fixed-point iteration on $\mathbf{F}_0$ to let it converge to $\mathbf{F}^{n+1}$. Specifically, we solve a sequence of optimization problems

$$\mathbf{v}^{n+1,j+1} = \text{argmin}_{\mathbf{v}} \frac{1}{2}\|\mathbf{v} - (\mathbf{v}^n + \mathbf{g}\Delta t)\|_{\mathbf{M}} + \sum_q V_q^0 \Psi_\theta(\mathbf{F}_q, \mathbf{F}_{0,q}^j, h_q^j), \quad \text{for } j = 0, 1, 2, ..., \tag{3.58}$$

treating the concatenated deformation gradients $\mathbf{F}_0^j$ and hardening states $\mathbf{h}$ as constants, which are only updated before each optimization as $\mathbf{F}_0^j = \mathbf{F}(\mathbf{v}^{n+1,j})$ and $\mathbf{h} = \mathbf{h}(\mathbf{F}_0)$. At convergence, we will obtain the true solution of Equation (3.49). In practice, a few number of fixed-point iterations can already generate high-quality results.

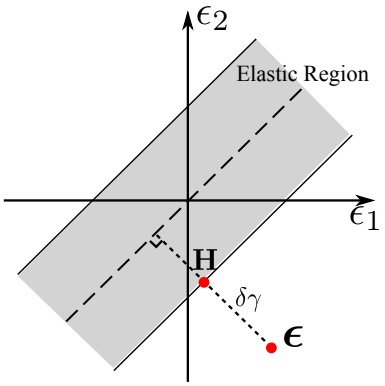


Figure 3.27: Volume-preserving projection.

Stability Regularizer We augment our learned potential with an extra quadratic regularizer to stabilize the simulation especially when the material is stiff or the time step size is large:

$$\Psi_\theta(\mathbf{F}, \mathbf{F}_0) = \mathcal{NN}_\theta(\mathbf{F}, \mathbf{F}_0) - (\nabla_{\mathbf{F}}\mathcal{NN}_\theta(\mathbf{F}_0, \mathbf{F}_0) - \boldsymbol{\tau}(\mathcal{Z}(\mathbf{F}_0))\mathbf{F}_0^{-\top}) \odot \mathbf{F} + \frac{1}{2}\mu\|\mathbf{F} - \mathbf{F}_0\|_F^2. \quad (3.59)$$

Here μ is the shear modulus of the material that Ψ_θ is learning. Note that this extra term is added after the model is trained instead of during the training. This extra term does not change the gradient at $\mathbf{F}_0$, so it will not change the fixed point of Procedure 3.58. Please see Section 3.2.5.3 for a comparison between simulations with and without this regularizer.

3.2.4.3 Learning Volume-Preserving Return Mapping

The return mapping $\mathcal{Z}$ required by PlasticityNet can be either given analytically or learned. Note that with different combinations of many practical elasticity and plasticity models, the return mapping may not have a closed-form solution, and the projection can only be performed by solving a nonlinear system of equations.

Here we provide a simple approach to learn a volume-preserving return mapping, which ensures that $\det(\mathcal{Z}(\mathbf{F})) = \det(\mathbf{F})$. For isotropic materials, the projection can be performed in the diagonal space, i.e., with $\mathbf{F} = \mathbf{U}\,\mathrm{Diag}(\boldsymbol{\Sigma})\mathbf{V}^\top$ being the singular value decomposition of $\mathbf{F}$; the projection is only needed for $\boldsymbol{\Sigma}$. In the diagonal space, a volume-preserving path is a straight line in the Hencky strain (defined as $\boldsymbol{\epsilon} = \log(\boldsymbol{\Sigma})$) space, which is perpendicular

to the diagonal line. The direction of the projection path is $\hat{\boldsymbol{\epsilon}} = \boldsymbol{\epsilon} - \text{sum}(\boldsymbol{\epsilon})\mathbf{1}$. The volume-preserving projection in the Hencky strain can be unified by $\mathbf{H} = \boldsymbol{\epsilon} - \delta\gamma \frac{\hat{\boldsymbol{\epsilon}}}{\|\hat{\boldsymbol{\epsilon}}\|}$ for some $\delta\gamma$, with $\mathcal{Z}^{\Sigma}(\boldsymbol{\Sigma}) = \exp(\mathbf{H})$ and $\mathcal{Z}(\mathbf{F}) = \mathbf{U}\,\text{Diag}(\mathcal{Z}^{\Sigma})\mathbf{V}^{\top}$. An illustration is shown in Figure 3.27.

The elastic region is usually represented by an implicit function $y(\boldsymbol{\Sigma}) \leq 0$. We can use a neural network to predict $\delta\gamma$, where the training leverages the differentiability of the implicit representation for the elastic region boundary. The volume-preserving path usually has two intersections with the elastic region boundary. To eliminate this ambiguity, we clamp the output of the neural network with a maximum $\|\hat{\boldsymbol{\epsilon}}\|$. We define our neural-network-based return mapping on the diagonal space as:

$$\delta\gamma_\theta(\boldsymbol{\Sigma}) = \min\{\mathcal{NN}_\theta(\Sigma), \|\hat{\boldsymbol{\epsilon}}\|\}, \quad \mathcal{Z}^{\Sigma}_\theta(\boldsymbol{\Sigma}) = \begin{cases} \exp(\boldsymbol{\epsilon} - \delta\gamma_\theta \frac{\hat{\boldsymbol{\epsilon}}}{\|\hat{\boldsymbol{\epsilon}}\|}), & y(\boldsymbol{\Sigma}) > 0, \\ \boldsymbol{\Sigma}, & y(\boldsymbol{\Sigma}) \leq 0. \end{cases} \tag{3.60}$$

The training loss function for a single $\boldsymbol{\Sigma}$ is defined as

$$\mathcal{L}(\boldsymbol{\Sigma}; \theta) = \begin{cases} y(\mathcal{Z}^{\Sigma}_\theta(\boldsymbol{\Sigma}))^2 + \max\{\delta\gamma_\theta(\boldsymbol{\Sigma}) - \|\hat{\boldsymbol{\epsilon}}\|, 0\}, & y(\boldsymbol{\Sigma}) > 0 \\ 0, & y(\boldsymbol{\Sigma}) \leq 0 \end{cases} \tag{3.61}$$

Here, the first term is to pull the points outside the elastic region back onto the boundary. The second term is to avoid these points to be always projected onto the diagonal due to the clamping in $\delta\gamma_\theta$. To account for hardening, we only need to let the $\delta\gamma$ network accept an extra hardening state variable h: $\delta\gamma_\theta(\boldsymbol{\Sigma}, h) = \min\{\mathcal{NN}_\theta(\Sigma, h), \|\hat{\boldsymbol{\epsilon}}\|\}$. The learned return mapping is then ready to be used by our PlasticityNet.

3.2.5 Experiments

We show examples to demonstrate the capability of our PlasticityNet in learning versatile plasticity models. Our physical simulators are implemented using C++, and we applied PyTorch to learn the potential energies, which are then loaded into our simulators with TorchScript. All our potential energies are multilayer perceptrons (MLPs) with Swiss

Table 3.3: Network Architectures and Training Details

Model	**MLP layers**
Sand (StVK+Drucker-Prager)	[8,32,32,32,1]
Snow (Neohooken+NACC)	[9,32,32,32,1]
Metal (StVK+von-Mises)	[9,32,32,32,1]
Sand 3D	[18,64,64,64,1]
Metal 3D (StVK+von-Mises)	[19,64,64,64,1]
Snow 3D	[19,64,64,64,1]

activation functions ($x\,\text{sigmoid}(x)$) except the output layer. They are all trained using the Adam optimizer (Kingma and Ba, 2014) on a single Nvidia RTX 3090 GPU with the same parameters: initial learning rate $\alpha = 0.01$, decay rate $\gamma = 0.95$, decay step 1000. The training data is generated during the training process with random sampling, and the batch size is 2^{16} for all cases. The models are all trained with 20000 epochs. The detailed architectures for each model is listed in Table 3.3. All ground-truth data are generated using standard explicit time integration with analytical plasticity returning mapping under small time step sizes for stability and accuracy. With our PlasticityNet, we can robustly simulate elastoplastic behaviors with much larger time step sizes using optimization time integrators.

3.2.5.1 Training

The training of PlasticityNet only requires the return mapping (either given analytically or pre-trained) for the plasticity model and the Kirchhoff stress for the underlying elasticity model. There is no need for extra labeled data. At each epoch, we will sample a new batch of $(\mathbf{F}, \mathbf{F}_0, \mathbf{h})$. The sampling of deformation gradients is based on its singular value decomposition $\mathbf{F} = \mathbf{U}\,\text{Diag}(\mathbf{\Sigma})\mathbf{V}^\top$, with $\mathbf{U}, \mathbf{V}$ being two rotation matrices. To sample $\mathbf{F}$ and $\mathbf{F}_0$ so that their singular values are close to each other, we set $\mathbf{F}_0 = \mathbf{R}_1\,\text{Diag}(e^{\boldsymbol{\epsilon}})\mathbf{R}_2$ and $\mathbf{F} = \mathbf{R}_3\,\text{Diag}(e^{\boldsymbol{\epsilon}+\delta\boldsymbol{\epsilon}})\mathbf{R}_4$, where $\boldsymbol{\epsilon}$ is a randomly sampled vector, $\delta\boldsymbol{\epsilon}$ is a random perturbation, and $\mathbf{R}_i$'s are randomly sampled rotation matrices. The hardening state is sampled uniformly from an appropriate range depending on the plasticity model. In this work, we uniformly sample $\boldsymbol{\epsilon}$ from $[-1, 1]^d$, $\delta\boldsymbol{\epsilon}$ from $[-0.1, 0.1]^d$ for sand plasticity and metal plasticity, and $[-0.2, 0.2]^d$ for the snow plasticity. The training loss curves of our 2D models are shown in

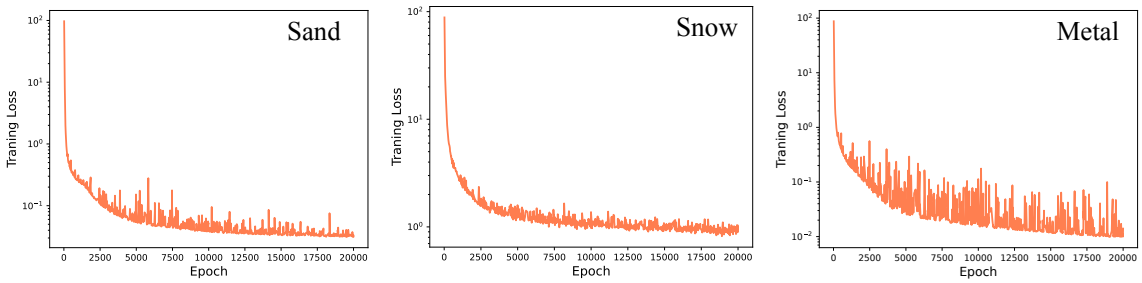
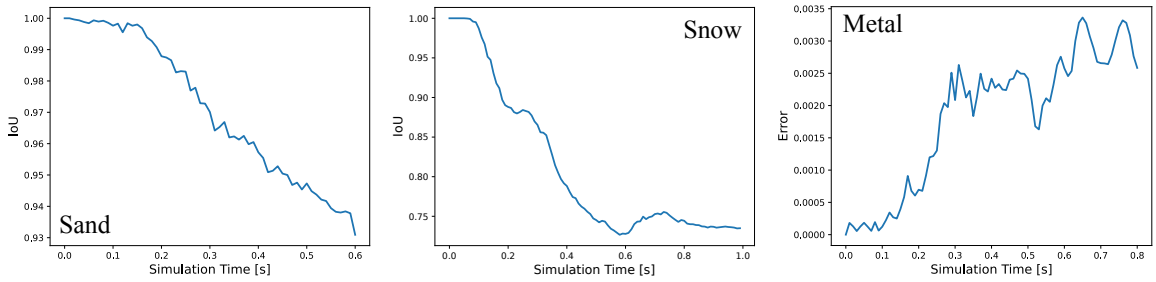


Figure 3.28: Training losses of our 2D models.



(a) (b) (c)

Figure 3.29: **(a) (b)** The intersection-over-union (IOU) (Huang et al., 2021) measure between the ground truth and our results. The IOUs are computed using the mass distributions on the MPM grid. **(c)** The average FEM nodal position difference. Note that the bounding box of the 2D metal frame is $0.1m \times 0.18m$.

Table 3.4: Computational costs of 2D experiments.

Experiment	Ours		Explicit	
	Time step (s)	s/frame	Time step (s)	s/frame
Sand	1e-3	12.58	1e-5	**6.20**
Snow	1e-3	35.56	1e-5	**6.78**
Von-Mises Metal	1e-2	**1.08**	1e-5	5.39
Neohookean Metal	1e-2	**1.03**	1e-5	7.88
MPM-FEM Coupling	1e-3	**38.90**	1e-6	184.58

Figure 3.28.

3.2.5.2 Testing on 2D Simulations

In this section, explicit time integrators are used to generate the ground-truth data for the validation of the optimization time integrators with PlasticityNet on multiple 2D experiments. The quantitative comparisons are plotted in Figure 3.29. We additionally include the computational costs in Table 3.4. We remark that the main objective of our work is not

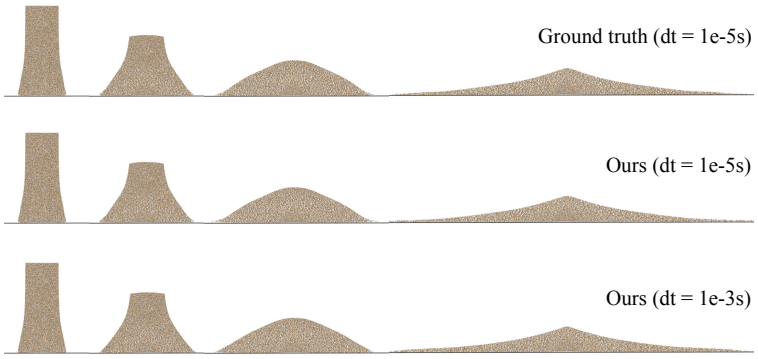


Figure 3.30: Sand plasticity.

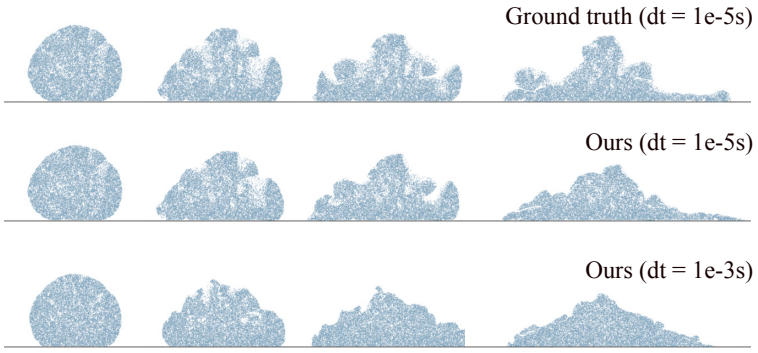


Figure 3.31: Snow plasticity with hardening.

to surpass the performance of the existing simulation of every constitutive model, but to provide a methodology that enables the usage of implicit plasticity in an optimization time integration framework.

Sand Plasticity We start by learning the elastoplastic model of dry sand (Figure 3.30). The model combination is St. Venant-Kirchhoff (StVK) elasticity, and the closed-form Drucker Prager plasticity return mapping (Klár et al., 2016) (See Appendix B.2.1). In this example, we simulate a column of sand falling onto the ground under gravity with MPM. Our method generates visually identical results compared to the ground truth, both with the same time step size and a 100× larger time step size. The quantitative comparison between our results and the ground truth is shown in Figure 3.29a. Note that there is no hardening mechanism in this plasticity model, so our PlasticityNet does not need the hardening state in its input.

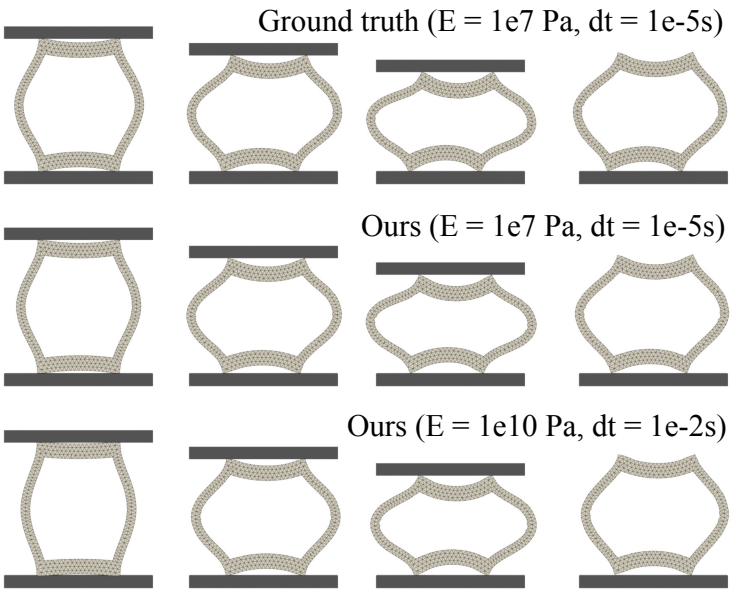


Figure 3.32: Metal plasticity with hardening.

Snow Plasticity with Hardening Snow is an elastoplastic material that can become stiffer under compression. Essentially, this is the effect of hardening where its elastic region get expanded. The variation in the stiffness across the snow body makes it easily fracture. Here we simulate a snowball hitting the ground in the MPM simulator (Figure 3.31). We use Neo-Hookean elasticity with the closed-form non-associative Cam-Clay plasticity return mapping (Gaume et al., 2018) (See Appendix B.2.2). Our method generates similar results compared to the ground truth when using the same time step size. The quantitative comparison of our results and the ground truth is shown in Figure 3.29b. Our framework remains stable even under much larger time step sizes. However, more numerical damping artifacts are introduced as the time step size increases, which results in slightly different behaviors compared to the ground truth.

Metal Plasticity with Hardening Metal is another common plastic material with hardening. In this example, we train PlasticityNet to learn metal plasticity with the StVK elasticity and the closed-form von-Mises plasticity return mapping (Mises, 1913a). (See Appendix B.2.3) We simulate a metal frame compressed by a rigid plate in the FEM simulator (Figure 3.32), where the Incremental Potential Contact (IPC) (Li et al., 2020) is used to

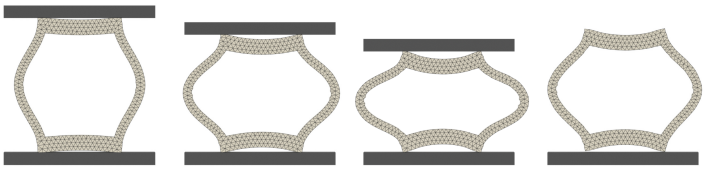


Figure 3.33: Learned metal plasticity return mapping with neo-Hookean elasticity.

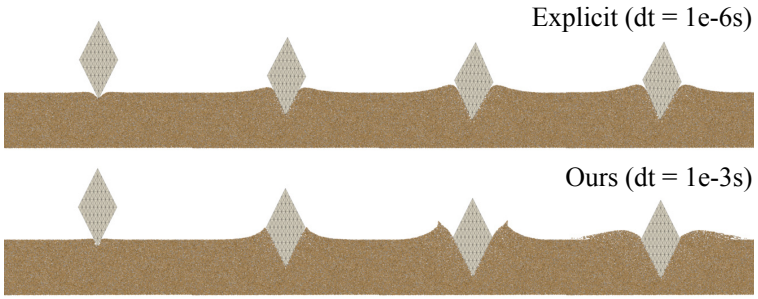


Figure 3.34: Two-way coupling between FEM elasticity and MPM sand plasticity.

handle the frictional contact between the solids. When we run the explicit time integration to generate ground truth, we have to decrease Young's modulus to enable using large enough time step sizes so that the simulation can be finished in practical time. Our method with the original setting generates visually identical results using a much large time step size. The quantitative comparison of our result and the ground truth is shown in Figure 3.29c.

Metal Plasticity Return Mapping Here we show an example simulated using PlasticityNet with a learned von-Mises plasticity return mapping. The underlying elasticity is neo-Hookean, instead of the StVK model in the last example (See Appendix B.2.4). Note that for Neo-Hookean material, there is no closed-form solution available for the von-Mises return mapping. In this case, a nonlinear optimization problem will need to be solved to perform the return mapping for every element/particle in every time step, which could severely slow down the standard explicit time integration. Using the same parameters as the metal compression experiments above, we show that PlasticityNet with learned plasticity return mapping under neo-Hookean elasticity can generate qualitatively similar results (Figure 3.33) to those from PlasticityNet with closed-form return mapping under the StVK elasticity.

MPM-FEM Coupling PlasticityNet enables the simulation of plastic materials in the MPM-FEM coupling framework BFEMP (Li et al., 2021c), where only pure elasticity was supported. When simulating with explicit BFEMP, the time step size required by stability is the minimum between MPM step size upperbound and FEM step size upperbound. Here we show an example where a stiff FEM elastic body falls onto MPM sand (Figure 3.34), where the implicit BFEMP can use a time step size 1000x larger than the explicit BFEMP and achieves an approximately 5x speedup in wall-clock time. We also remark that when the time step size is small (as is required to keep the explicit time integration stable in this case), MPM suffers from excessive numerical damping due to the significant amount of particle-grid transfers. This is a known issue of explicit MPM simulations.

Different Energy Representations Here we include some different energy representations we investigated (Figure 3.35), whose inaccurate results motivated us to develop our final representation Equation (3.56). These experiments are all conducted on the 2D sand column collapse example. The first straightforward idea is to find a globally defined neural energy function $\Psi(\mathbf{F}) = \Psi_\theta(\mathbf{F})$ that solves Equation (3.54), where θ is the parameter of the neural network. Note that it is theoretically unachievable to train a global potential energy function because the right hand side of Equation (3.54) is not integrable in the plastic region. But it is still worth trying to explore an approximation by minimizing $\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{F}} \left\| \frac{\partial \Psi_\theta}{\partial \mathbf{F}}(\mathbf{F}) - \boldsymbol{\tau}(\mathcal{Z}(\mathbf{F}))\mathbf{F}^{-\top} \right\|_F^2$. However, the experiment shows that this formulation makes the sand column behave like an elastic body. It is also noteworthy that the sand column cannot even maintain the rest shape at the first frame: it erroneously shrinks suddenly and jumps off the ground. Additional insight is provided by realizing that a linear correction is necessary to exactly vanish stress when the deformation gradient is the identity; so we experiment with $\Psi(\mathbf{F}) = \Psi_\theta(\mathbf{F}) - \nabla_{\mathbf{F}}\Psi_\theta(\mathbf{I})$. This formulation unfortunately also leads to an insufficient capture of plasticity, giving an elastic and visually distinct incorrect result. These observations motivate us to investigate a family of potential energies to solve Equation (3.54) locally. We first use $\Psi(\mathbf{F}, \mathbf{F}_0) = \Psi_\theta(\mathbf{F}, \mathbf{F}_0) - \nabla_{\mathbf{F}}\Psi_\theta(\mathbf{I}, \mathbf{F}_0)$ and train with the loss function in Equation (3.57). The simulation captures certain plastic behaviors when the deformation is

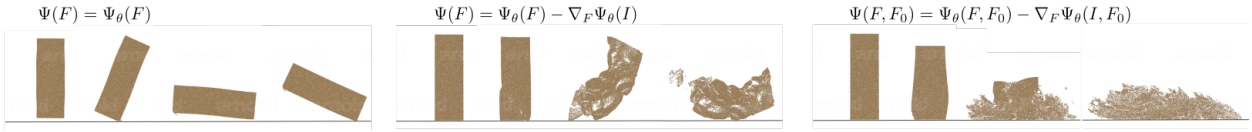


Figure 3.35: Ablation studies on different energy representations.

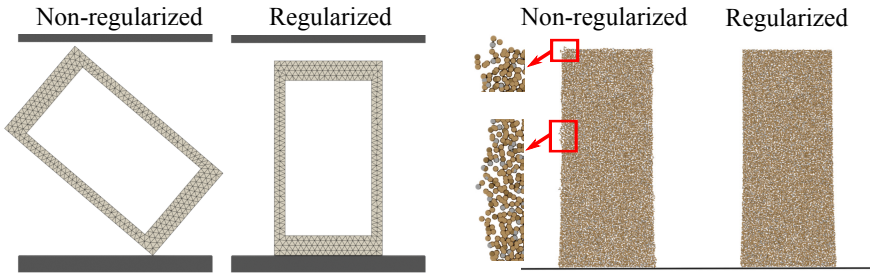


Figure 3.36: The regularizer significantly improves the stability of the simulation.

small, but the result quickly deviates from the ground truth when the deformation becomes larger. Finally, we come up with Equation (3.56) to achieve the nice results in Figure 3.30.

3.2.5.3 Ablation Studies

Stability Regularizer As an ablation study for the stability regularizer in Equation (3.59), we compare the simulations with and without the regularizer on two 2D examples (Figure 3.36). Without the regularizer, the metal frame can not even stay in its original rest configuration after the first time step. In the sand example, particles in the highlighted regions tend to separate from the sand column in a non-physical manner. These demonstrate that our regularizer significantly improves the stability of the simulation.

3.2.5.4 Testing on 3D Simulations

Extending PlasticityNet to support 3D simulation is straightforward. We only need to increase the dimension of the inputs to the PlasticityNet. To improve the expressiveness of the network, we also increase the dimension of hidden variables. Here we demonstrate the 3D versions of our 2D examples with similar physical parameters in Figure 3.37: 3D sand plasticity, 3D snow plasticity, and 3D metal plasticity. The 3D metal is simulated with $\Delta t = 10^{-2}$, and for sand and snow, we use $\Delta t = 10^{-3}s$ to satisfy the CFL condition (Courant

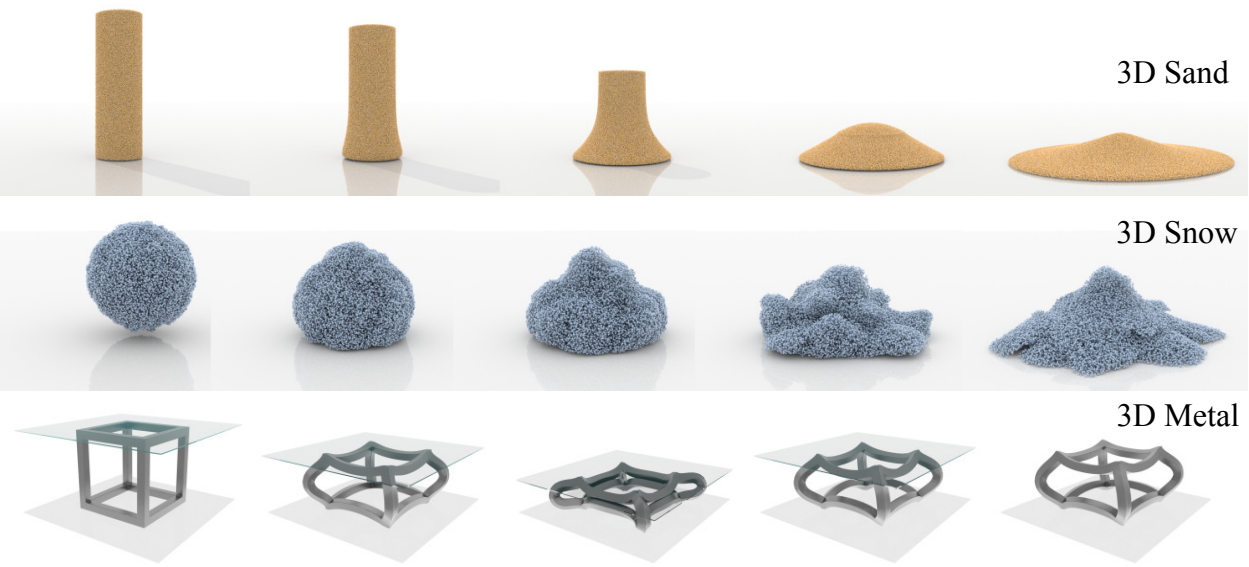


Figure 3.37: 3D simulations with sand plasticity, snow palsticity and metal plasticity.

et al., 1967) in MPM, preventing the particles from traveling farther than the grid cell spacing in a single time step.

3.2.6 Conclusion

We proposed PlasticityNet, a neural network-based elastoplastic model learning framework that is agnostic to spatial discretizations. PlasticityNet represents the elastoplastic forces as the positional gradients of learned potential energies, so that optimization time integration could be applied to achieve robust and efficient simulation at large time step sizes. We demonstrated that low-level components in traditional physical simulation frameworks can be substituted with neural networks to obtain desired numerical properties that benefit the computation. Notably, this also avoids tedious analytical derivations or expensive nonlinear root-findings without significantly sacrificing the accuracy. We believe our work can inspire more research that applies machine learning to physical simulation in the bottom-up style, maintaining fundamental physical properties and applicability to general scenarios.

Limitations and Future Work There are several limitations of our framework. **(1)** We cannot guarantee our fixed-point iteration will converge for arbitrary scenes. It is theoretically

valuable to explore under what conditions the fixed-point can converge and what loss functions can accelerate the convergence. **(2)** Although the regularizer added during the simulation improves the stability of the simulation without changing the solution at convergence, it may introduce some artificial viscosity because the regularized energy penalizes deformations away from $\mathbf{F}_0$. Running more fixed-point iterations can alleviate this issue. It will also be interesting to explore adaptive weighting mechanisms for the regularizer, or convert this soft regularizer into a hard constraint. **(3)** We do not consider the Hessian of the learned plastic energy in our training. Since we use second-order methods to perform optimization time integration, the properties of the Hessian matrices may have an impact on the convergence of the optimization method. Although the Jacobian matrices of the target gradients are asymmetric, it may be helpful if the Hessian of our learned elastoplastic energy can approximate them so that the stiffness of the material can be more accurately resolved. **(4)** Principled physical assumptions of the learned potential energies by PlasticityNet, such as lower-boundedness and convexity, are not enforced. It is interesting to explore whether enforcing these energy properties would positively influence the convergence of the optimizations and fixed-point iterations. **(5)** A trained PlasticityNet can be directly re-scaled to accommodate a different Young's modulus, but it needs to be re-trained for materials with different Poisson's ratio or plasticity parameters. It is an important future work to let our model more easily generalize to different parameters. For example, these parameters can become extra inputs to the neural network. The generalized energy can also be integrated into differentiable simulators (Hu et al., 2020; Qiao et al., 2021a) to solve many inverse problems (Ma et al., 2021; Lin et al., 2022a; Li et al., 2022c).

CHAPTER 4

FEM-MPM Coupled Simulation

4.1 BFEMP: Interpenetration-Free MPM-FEM Coupling with Barrier Contact

4.1.1 Introduction

The Material Point Method (MPM) (Sulsky et al., 1995; de Vaucorbeil et al., 2019) extends the Particle-In-Cell (PIC) (Harlow, 1962) and the Fluid Implicit Particle (FLIP) (Brackbill et al., 1988) methods from fluid dynamics to computational solids. In contrast to the commonly used Total Lagrangian Finite Element Method for elastodynamics (Hughes, 2012), MPM utilizes Lagrangian particles to represent continuum materials and an Eulerian background grid to discretizes the governing equations. Except for recent advancements in Total Lagrangian MPM (de Vaucorbeil et al., 2020; de Vaucorbeil and Nguyen, 2021), MPM is usually considered to be following an Updated Lagrangian kinematic assumption with particles tracking historical deformation, strain, stress, and other constitutive variables through evolving them with velocity fields. The hybrid Lagrangian-Eulerian perspective combined with the Updated Lagrangian kinematics puts MPM in a very advantageous position in modeling and simulating high-speed, large-deformation, and topologically changing events (Zhang et al., 2016c). Having gained a lot of attention in the last two decades, MPM and its variants (Bardenhagen and Kober, 2004; Zhang et al., 2011; Sadeghirad et al., 2011; Gan et al., 2018; Hu et al., 2018a) have been successfully applied in challenging problems including multiphase flows, fracture, contact, adaptivity, free-surface flows, soil-fluid mixture, explosives, and granular media (Zhang et al., 2008; Homel and Herbold, 2017; Nairn, 2003; Homel and Herbold, 2018; Tan and Nairn, 2002; Zhang et al., 2017a; Abe et al., 2014; Guilkey et al., 2007; Gaume et al.,

2018).

Although MPM has been demonstrated effective on a wide range of materials, many application scenarios favor other discretization schemes due to considerations in efficiency, accuracy, and suitability. Correspondingly, a large number of engineering applications necessitate hybrid or coupled solvers combining MPM with other discretization choices. For example, MPM has been coupled or hybridized with the Discrete Element Method (DEM) for solid-fluid interaction (Yang et al., 2017b) and granular media (Liu et al., 2017a; Jiang et al., 2020b; Chen et al., 2021b), the Finite Difference Method (FDM) for multiphase saturated soils (Higo et al., 2010), and the Smoothed Particle Hydrodynamics (SPH) for solid mechanics (Raymond et al., 2018).

Even though MPM itself can be derived as a Galerkin Finite Element Method, it is still more common in computational solid mechanics to use the term "FEM" to refer to the standard Total Lagrangian mesh-based FEM discretization. In this sense, MPM is much less developed than FEM and still suffers from unique challenges in aspects such as stability, accuracy, boundary condition enforcement, and numerical fracture (Cummins and Brackbill, 2002; Guilkey and Weiss, 2003; Homel et al., 2016). Therefore, FEM is often more suitable for analyzing hyperelastic structures under small or moderate deformations. Resultingly, MPM-FEM coupling becomes highly desirable in many multi-material simulation tasks or those involving strongly heterogeneous deformations, for example, the blast event simulations involving vehicles (Swensen et al., 2006). The coupling between MPM and FEM has been extensively studied over the last decade. A natural way to hybridize the two schemes is to treat FEM vertices as MPM particles and embed them into the MPM grid (Lian et al., 2011b; Jiang et al., 2015a). The FEM shell formulation can also be embedded into the MPM grid (Banerjee, 2005). In a similar fashion, EMPFE (Zhang et al., 2006) discretizes the entire domain according to the severity of deformation – small deformation regions with FEM, while large deformation regions with MPM, and then it embeds the displacement of interface FEM vertices to the MPM grid. Despite its simplicity, additional treatment for eliminating the hourglass mode is necessary due to simple trilinear MPM kernels for the interface computation. Later on, AFEMP (Lian et al., 2012) extends this idea to support the dynamical conversion

from severely distorted FEM elements to MPM particles. These methods handle material interactions through the grid-based MPM contact and inherit common MPM-based contact characteristics such as the strict nonslip condition between contacting interfaces.

To circumvent these issues, CFEMP (Lian et al., 2011a) was proposed to only use the interface MPM grid for contact detection while computing frictional contact forces directly based on the contact conditions. CFEMP has been successfully applied to coupling FEM membranes with MPM solids (Lian et al., 2014), and also to modeling needle-tissue interactions (Li et al., 2021b). However, these grid-based MPM-FEM coupling strategies require the FEM boundary element size to be similar to the MPM grid spacing. If it is too large, interpenetration can happen, while if it is too small, intrinsic damping will appear (Lian et al., 2011b), and the time step size for explicit time integration would also be more restricted. Accordingly, Cheon and Kim (Cheon and Kim, 2018) proposed to add extra distributed interaction (DI) nodes on the FEM boundary elements to improve contact detection for large-sized elements.

An improvement to CFEMP that also applies to AFEMP was later proposed to couple FEM with MPM by handling contact primitive pairs between FEM boundary elements and nearby MPM particles (Chen et al., 2015; Wu et al., 2018). A penalty method is applied for computing the frictional contact forces. Later, Song *et al.* (Song et al., 2020) extended this idea with an iterative contact force computation approach for the simultaneous satisfaction of all contact conditions, together with an improved local search method to prevent interpenetration issues at the contact crack. Bewick (Bewick, 2004) proposed to insert intermediate nodes at the interface for 1D impact-resistant design problems, and the coupling forces are calculated by FEM displacements which are determined by MPM particles.

So far, all the discussed MPM-FEM coupling works are designed for explicit time integration. Imposing frictional contact between FEM and MPM within implicit time integration is challenging because the associated inequality constraints that need to be simultaneously enforced while solving the nonlinear system of equations are also nonlinear and non-smooth. Aulisa *et al.* (Aulisa and Capodaglio, 2019) proposed a monolithic coupling method for implicit MPM and FEM through a conforming interface mesh, while extra care is needed to

avoid the sticky artifact in receding contact cases. Larese *et al*. (Larese et al., 2019) discussed implicit MPM-FEM coupling researches in geomechanics. In a soil-structure interaction problem, the impact forces on the interface are transferred between the soil and the structure solver and iterated until convergence. A similar method for enforcing nonconforming boundary conditions for MPM (Chandra et al., 2021) was also applied. However, as pointed out in their work, the method requires smaller time increments for problems with high relative velocity towards the boundary, as otherwise, the boundary enforcement will be too late, and the incoming material points may penetrate the nonconforming boundary surfaces.

This paper explores MPM-FEM coupling under the assumption of implicit integration in both domains. Compared to explicit time integration, implicit schemes permit substantially larger time steps with superior stability for stiff nonlinear problems. Implicit MPM/GIMP has been explored by many researchers (Cummins and Brackbill, 2002; Guilkey and Weiss, 2001, 2003; Nair and Roy, 2012; Charlton et al., 2017; Love and Sulsky, 2006). We refer to these literature for more discussions about the advantages of implicit time integration. Our work follows the variational formulation of a wide family of implicit time integrators (Ortiz and Stainier, 1999; Kane et al., 2000), where the displacement evolvement in each time step is formulated as the stationarity point of a time discretized energy functional. The resulting optimization-based time integrator has been applied in MPM with Newton-Krylov (Gast et al., 2015), and more recently, quasi-Newton L-BFGS (Wang et al., 2020a) solvers.

In this work, integrating both MPM and FEM domains implicitly, we study MPM-FEM coupling based on contact mechanics (thus we do not consider objects with partially mixed discretization choices). A barrier-augmented variational frictional contact formulation is known as the Incremental Potential Contact (IPC) (Li et al., 2020; Li, 2020; Li et al., ND) was recently proposed for nonlinear elastodynamics with linear kernel FEM, which has also shown to be effective for codimensional models (Li et al., 2021a), reduced space dynamics (Ferguson et al., 2021b; Lan et al., 2021b), and embedded interfaces (Zhao et al., 2022). It formulates the contact problem during time stepping as minimizing a potential energy inside the manifold of interpenetration-free displacement trajectories characterized by boundary geometric primitives of elastic structures. Extending this approach, we model MPM-FEM

coupling as jointly finding optimal FEM mesh nodal displacements and MPM grid nodal displacements under the constraint that MPM particles, with their trajectories *embedded* in grid nodal displacements, maintain strict positive distances to the FEM mesh throughout the implicit integration. The resulting method is named barrier FEMP (BFEMP) because these constraints are enforced using *barrier* energies. Even though contact conditions are defined between MPM particles and the FEM mesh, the real displacement unknown variables for the MPM domain which the implicit time integration solves for are still defined on MPM grids. The MPM particles remain embedded quadrature points in the MPM grid degrees of freedom. Compared to soft penalty-based methods such as the particle-to-surface contact algorithm recently proposed by Nakamura *et al.* (Nakamura et al., 2021), BFEMP requires no stiffness parameter tuning and guarantees strict, hard non-penetration conditions under convergence. Another useful feature of the proposed coupling scheme is that it enables a new way of imposing irregular, separable, and frictional kinematic boundaries for MPM. Irregular boundaries for MPM is a recently advanced topic (Tjung et al., 2020). BFEMP inherently enables it by assigning all nodes in the FEM domain with prescribed Dirichlet displacements and letting MPM interacts with them.

4.1.2 Governing Equations

In this study we focus on elastodynamics based on continuum mechanics. The corresponding governing equations for a deformed continuum domain Ω^t with $\boldsymbol{x} \in \Omega^t$ and time $t \in [0, \infty)$ are given by (Bonet and Wood, 2008)

$$\frac{D\rho}{Dt} + \rho \nabla^{\mathbf{x}} \cdot \mathbf{v} = 0, \tag{4.1}$$

$$\rho \frac{D\mathbf{v}}{dt} = \nabla^{\mathbf{x}} \cdot \boldsymbol{\sigma} + \rho \mathbf{g}, \tag{4.2}$$

where $\rho(\mathbf{x}, t)$ is the density, $\mathbf{v}(\boldsymbol{x}, t)$ is the velocity, $\boldsymbol{\sigma}(\boldsymbol{x}, t)$ is the Cauchy stress, and $\mathbf{g}$ is the gravitational acceleration which is assumed to be the only body force. Under the finite strain assumption (as we shall assume throughout this paper), deformation $\phi(\mathbf{X}, t)$ maps $\mathbf{X} \in \Omega^0$ from the material space to $\mathbf{x} \in \Omega^t$ in the world space: $\boldsymbol{x} = \phi(\boldsymbol{X}, t)$. The deformation gradient

is defined as

$$\mathbf{F} = \frac{\partial \phi}{\partial \mathbf{X}}(\mathbf{X}, t) \tag{4.3}$$

to describe the local deformation.

In Lagrangian Finite Element analysis for nonlinear dynamics, it is often preferred to derive the weak form for $\boldsymbol{X} \in \Omega^0$. Here we can pull back the momentum equation to the material space. Denoting the first Piola-Kirchhoff stress $\mathbf{P} = \mathbf{P}(\mathbf{X}, t)$, the Lagrangian momentum equation is then

$$R_0 \mathbf{A}(\mathbf{X}, t) = \nabla^{\mathbf{X}} \cdot \mathbf{P}(\mathbf{X}, t) + R_0 \mathbf{g}, \tag{4.4}$$

where $R_0 = R(\mathbf{X}, 0)$ is the material density at time 0, and $\mathbf{A}(\mathbf{X}, t) = \frac{\partial^2 \Phi}{\partial t^2}(\mathbf{X}, t)$ is the Lagrangian acceleration, $\mathbf{V}(\boldsymbol{X}, t) = \frac{\partial \Phi}{\partial t}(\mathbf{X}, t)$ is the Lagrangian velocity. The Cauchy stress is related to the first Piola Kirchhoff stress as

$$\boldsymbol{\sigma} = \det(\boldsymbol{F})^{-1} \boldsymbol{P} \boldsymbol{F}^T. \tag{4.5}$$

In this paper we are concerned with hyperelasticity. Thus there exists an elastic energy density function $\psi(\boldsymbol{F})$ such that

$$\boldsymbol{P}(\boldsymbol{F}) = \frac{\partial \psi}{\partial \boldsymbol{F}}(\boldsymbol{F}). \tag{4.6}$$

Without loss of generality, we focus on isotropic materials and adopt a compressible Neo-Hookean constitutive model with

$$\psi(\boldsymbol{F}) = \frac{\mu}{2} \mathrm{tr}(\boldsymbol{F}^T \boldsymbol{F} - \boldsymbol{I}) - \mu \log(J) + \frac{\lambda}{2} \log(J)^2, \tag{4.7}$$

where $J = \det \boldsymbol{F}$, μ and λ are the Lamé parameters.

Remark 4.1.1. Many hyperelasticity models such as the Neo-Hookean model are only well defined for $J > 0$. Discretely this will impose a nonlinear strict inequality constraint on

the displacements for each quadrature point with a discrete sample of $\boldsymbol{F}$. Ignoring these constraints in a nonlinear optimization-based Newton solver will cause floating-point number failures when a search step tries to evaluate energy or stress quantities at an intermediate configuration with $J \leq 0$. Instead of requiring a reduction of the time step, we directly enforce this constraint through a line search filtering strategy (see Section 4.1.5).

4.1.2.1 Weak Form

Given trial function $\boldsymbol{Q}(\cdot, t) : \Omega^0 \to \mathbb{R}^3$, the corresponding *Lagrangian* weak form of Equation (4.4) is

$$\int_{\Omega^0} Q_\alpha(\boldsymbol{X}, t) R_0 A_\alpha(\boldsymbol{X}, t) d\boldsymbol{X} = \int_{\partial\Omega^0} Q_\alpha T_\alpha dS(\boldsymbol{X}) - \int_{\Omega^0} Q_{\alpha,\beta} P_{\alpha\beta} d\boldsymbol{X} + \int_{\Omega^0} Q_\alpha(\boldsymbol{X}, t) R_0 g_\alpha d\boldsymbol{X}, \tag{4.8}$$

where $T_\alpha = P_{\alpha\beta} N_\beta$ (with $\boldsymbol{N}(\boldsymbol{X})$ being the material space normal) is the traction field at the domain boundary $\partial\Omega^0$, on which one could presribe traction boundary conditions as needed.

While Total Langrangian FEM typically discretizes Equation (4.8), MPM usually adopts the Updated Lagrangian view and consequently discretizes an *Eulerian* weak form instead (Zhang et al., 2016c). Correspondingly the stress derivatives are discretized at the current configuration Ω^t. We can either push forward Equation (4.8) or directly integrate Equation (4.2) to reach

$$\int_{\Omega^t} q_\alpha(\boldsymbol{x}, t) \rho(\boldsymbol{x}, t) a_i(\boldsymbol{x}, t) d\boldsymbol{x} = \int_{\partial\Omega^t} q_\alpha t_\alpha ds(\boldsymbol{x}) - \int_{\Omega^t} q_{\alpha,\beta} \sigma_{\alpha\beta} d\boldsymbol{x} + \int_{\Omega^t} q_\alpha(\boldsymbol{x}, t) \rho(\boldsymbol{x}, t) g_\alpha d\boldsymbol{x}, \tag{4.9}$$

where $\boldsymbol{q}(\boldsymbol{x}, t) = \boldsymbol{Q}(\Phi^{-1}(\boldsymbol{x}, t), t)$ is the push forward of $\boldsymbol{Q}$, *i.e.*, an Eulerian trial function, $\boldsymbol{a}(\boldsymbol{x}, t) = \boldsymbol{A}(\Phi^{-1}(\boldsymbol{x}, t), t) = \frac{D\boldsymbol{v}}{Dt}(\boldsymbol{x}, t) = \frac{\partial \boldsymbol{v}}{\partial t} + \boldsymbol{v} \cdot \nabla \boldsymbol{v}$, and $\boldsymbol{t} = \boldsymbol{\sigma}\boldsymbol{n}$ is the traction at $\partial\Omega^t$ with $\boldsymbol{n}(\boldsymbol{x})$ being the outward pointing normal.

4.1.2.2 Incremental Variational Form

To enable the development of efficient optimization-based time integrators, we follow the variational treatment of the time-discretized incremental problem (Ortiz and Stainier, 1999; Radovitzky and Ortiz, 1999). Concretely, for a broad family of time discretization schemes (we will focus on backward Euler and the Newmark-β family in this paper), the solution of Equation (4.8) during an incremental time step, *i.e.*, the advancement from $\phi^n = \phi(\boldsymbol{X}, t^n)$ to $\phi^{n+1} = \phi(\boldsymbol{X}, t^{n+1})$), for an hyperelastic material is given by minimizing the following functional:

$$I(\phi^{n+1}) = \int_{\Omega_0} \left(\frac{1}{2} \frac{R_0}{\beta \Delta t^2} \|\phi^{n+1}\|^2 + 2\alpha \Psi(\boldsymbol{F}^{n+1}) \right) d\boldsymbol{X} - \int_{\Omega_0} R_0 \bar{\boldsymbol{B}}_{n+1} \cdot \phi^{n+1} d\boldsymbol{X} - 2\alpha \int_{\partial\Omega_0} \boldsymbol{T} \cdot \phi^{n+1} dS(\boldsymbol{X}), \tag{4.10}$$

where

$$\bar{\boldsymbol{B}}_{n+1} = 2\alpha \mathbf{g} + \frac{1}{\beta \Delta t^2} \left(\phi^n + \Delta t \frac{\partial \phi}{\partial t}(\boldsymbol{X}, t^n) + \alpha \left(1 - 2\beta\right) \Delta t^2 \frac{\partial^2 \phi}{\partial t^2}(\boldsymbol{X}, t^n) \right) \tag{4.11}$$

encodes the inertia term and is a constant field in the minimization problem. The velocity update rule is

$$\frac{\partial \phi}{\partial t}(\boldsymbol{X}, t^{n+1}) = \frac{\partial \phi}{\partial t}(\boldsymbol{X}, t^n) + \Delta t \left(\left(1 - \gamma\right) \frac{\partial^2 \phi}{\partial t^2}(\boldsymbol{X}, t^n) + \gamma \frac{\partial^2 \phi}{\partial t^2}(\boldsymbol{X}, t^{n+1}) \right). \tag{4.12}$$

Here we have slightly modified the energy proposed by Radovitzky and Ortiz (Radovitzky and Ortiz, 1999) to let it be compatible with backward Euler since their version was explicitly built for Newmark's algorithm.

Note that in the case of backward Euler ($\alpha = 1$, $\beta = \frac{1}{2}$, $\gamma = 1$), it can be easily shown that the Euler-Lagrangian equation of functional (4.10) gives back the time-discretized momentum balance

$$\frac{R_0}{\Delta t^2} \phi^{n+1} - \nabla^{\boldsymbol{X}} \cdot \boldsymbol{P}^{n+1} = R_0 \mathbf{g} + \frac{R_0}{\Delta t^2} (\phi^n + \Delta t \boldsymbol{V}^n). \tag{4.13}$$

Similarly, for Middle-point Newmark ($\alpha = \frac{1}{2}$, $\beta = \frac{1}{4}$, $\gamma = \frac{1}{2}$), we would get

$$\frac{R_0}{\Delta t^2}\phi^{n+1} - \frac{1}{4}\nabla^{\boldsymbol{X}} \cdot \boldsymbol{P}^{n+1} = \frac{1}{4}R_0\mathbf{g} + \frac{R_0}{\Delta t^2}(\phi^n + \Delta t\boldsymbol{V}^n + \frac{1}{4}\Delta t^2\boldsymbol{A}^n). \tag{4.14}$$

Both of them match the results from temporally discretizing the Lagrangian form of Equation (4.2).

4.1.2.3 Coupling

In the proposed framework, the Lagrangian form is discretized on the FEM domain Ω_F (Section 4.1.3.1), while the Eulerian form is discretized on the MPM domain Ω_M (Section 4.1.3.2). We assume

$$\Omega_F^0 \cap \Omega_M^0 = \varnothing \tag{4.15}$$

and their corresponding deformation map $\boldsymbol{x}_M = \phi_M(\boldsymbol{X}_M, t)$ and $\boldsymbol{x}_F = \phi_F(\boldsymbol{X}_F, t)$ are fully governed by their own variational forms in the absence of any coupling mechanism. In other words, without coupling, evolving the two domains independently (with two solvers) is equivalent to minimizing

$$\Pi(\phi_M, \phi_F) = I(\phi_M) + I(\phi_F) \tag{4.16}$$

by directly letting $\Omega = \Omega_F \cup \Omega_M$.

The coupling between FEM and MPM domains are then modeled via imposing the non-interpenetration constraints:

$$\Omega_F^t \cap \Omega_M^t = \varnothing, \quad \forall t \in [0, \infty) \tag{4.17}$$

between the two domains. Note that the time is discretized with $t = t^0, t^1, t^2, \ldots, t^n$. We use Ω^n to denote Ω^{t^n} for notational simplicity. See Figure 4.1 for an illustration. Thus the final variational MPM-FEM coupling problem can be described as minimizing Equation (4.16)

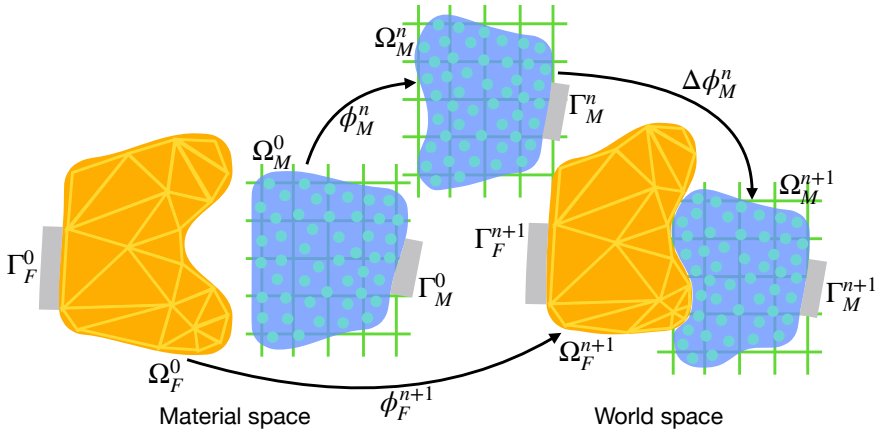


Figure 4.1: **Deformation map.** The material space of FEM and MPM domains (Ω_{F}^0 and Ω_{M}^0) on the left are mapped via ϕ_{F} and ϕ_{M} to the world space ($\Omega_{\mathrm{F}}^{n+1}$ and $\Omega_{\mathrm{M}}^{n+1}$) on the right, with $\Omega_{\mathrm{F}}^t \cap \Omega_{\mathrm{M}}^t = \varnothing$ for all $t \in [0, \infty)$. With Updated Lagrangian kinematics, MPM treats Ω_M^n as the "intermediate" material space and focuses on studying the deformation from Ω_M^n to Ω_M^{n+1}. Here Γ represents Dirichlet boundary or nonzero Neumann boundary.

under the equality constraint described by Equation (4.17).

The feasible region described by Equation (4.17) can be equivalently expressed as

$$\phi_M(\boldsymbol{X}_M, t) \neq \phi_F(\boldsymbol{X}_F, t), \quad \forall \boldsymbol{X}_M \in \Omega_M^0, \boldsymbol{X}_F \in \Omega_F^0, t \in [0, \infty). \tag{4.18}$$

Moreover, if we define

$$d(\phi_M, \phi_F, t) = \min_{\boldsymbol{X}_F, \boldsymbol{X}_M} \|\phi_F(\boldsymbol{X}_F, t) - \phi_M(\boldsymbol{X}_M, t)\| \tag{4.19}$$

to describe the Euclidean proximity between Ω_M^t and Ω_F^t, Equation (4.18) can be further converted to a strict inequality constraint

$$d(\phi_M, \phi_F, t) > 0, \quad \forall t \in [0, \infty). \tag{4.20}$$

Note that in Equation (4.15), we have assumed that the undeformed domains are non-overlapping. Thus the minimization problem starts with a strictly feasible solution at $t = 0$.

In Section 4.1.4.1 we describe a barrier method that results in a contact pressure for enforcing Equation (4.15). See Section 4.1.4.3 for extra components on incorporating tangential

frictional effects.

4.1.3 Discretization

The finite element domain is discretized with linear simplex elements (triangles in 2D and tetrahedra in 3D), and the material point domain is discretized using a collection of material points and an Eulerian background grid with quadratic B-spline kernels. Both schemes adopt mass lumping and assume zero traction at boundaries unless otherwise specified in an example. Stacking all nodal positions, velocities, and accelerations from both the FEM mesh and the MPM grid at time step n as $x^n = [(x_{\mathrm{F}}^n)^T, (x_{\mathrm{M}}^n)^T]^T$, $v^n = [(v_{\mathrm{F}}^n)^T, (v_{\mathrm{M}}^n)^T]^T$, and $a^n = [(a_{\mathrm{F}}^n)^T, (a_{\mathrm{M}}^n)^T]^T$ where subscripts F stands for FEM while M for MPM, the unified time integration update rule can be written as

$$\begin{aligned} \tilde{v}^{n+1} &= v^n + \Delta t((1-\gamma)a^n + \gamma a^{n+1}), \\ \tilde{x}^{n+1} &= x^n + \Delta t v^n + \alpha \Delta t^2((1-2\beta)a^n + 2\beta a^{n+1})), \end{aligned} \tag{4.21}$$

and it is equivalent to first solving the optimization problem

$$\min_x : \left(\frac{1}{2}\|x - \hat{x}^n\|_M^2 + 2\alpha\beta\Delta t^2 \Psi(x)\right) \tag{4.22}$$

to get $\tilde{x}^{n+1}$, and then explicitly calculating $\tilde{v}^{n+1}$. Here $\hat{x}^n = x^n + v^n\Delta t + \alpha(1-2\beta)\Delta t^2 a^n$, and $\Psi(x) = \sum_q V_q^0 \psi(\boldsymbol{F}_q(x))$ with q belonging to FEM elements or MPM particles and V_q^0 the rest volume of a FEM element or a MPM particle.

For FEM, the time integration is solely performed on the Lagrangian nodal degrees of freedom throughout the simulation and so $\tilde{x}_{\mathrm{F}}^{n+1} = x_{\mathrm{F}}^{n+1}$ and $\tilde{v}_{\mathrm{F}}^{n+1} = v_{\mathrm{F}}^{n+1}$. But for MPM, Equations (4.21) are only part of the Eulerian time integration performed on the Eulerian grid before and after particle-grid transfers for x_{M} and v_{M}, so $\tilde{x}_{\mathrm{M}}^{n+1} \neq x_{\mathrm{M}}^{n+1}$ and $\tilde{v}_{\mathrm{M}}^{n+1} \neq v_{\mathrm{M}}^{n+1}$ (see Section 4.1.3.2 for details). Note that the minimization problem (4.22) is equivalent to the discrete form of the variational time integration in (Ortiz and Stainier, 1999; Li et al., 2019a; Wang et al., 2020a) for hyperelastic problems. x_{F} and x_{M} are coupled through contact

modeling between the two domains (Section 4.1.4).

4.1.3.1 The Finite Element Domain

For the FEM domain, nodal positions and velocities are stored on mesh vertices and updated directly. The nodal masses are kept constant, *i.e.*, $m_i^n = m_i$.

Inside any simplex element, the material and world space coordinate of an arbitrary location $\boldsymbol{X}$ are expressed using linear interpolation kernels $N_i(\boldsymbol{X})$ of node $\boldsymbol{X}_i$ as

$$\boldsymbol{X} = \sum_i N_i(\boldsymbol{X})\boldsymbol{X}_i \quad \text{and} \quad \phi(\boldsymbol{X}, t) = \sum_i N_i(\boldsymbol{X})\phi(\boldsymbol{X}_i, t). \tag{4.23}$$

Then according to the definition of $\boldsymbol{F}$ (Equation (4.3)), with N_i being the linear hat function, the deformation gradient is piecewise constant. Inside a linear simplex element e it is directly evaluated as a function of x:

$$\boldsymbol{F}_e(x) = \boldsymbol{T}_e(x)\boldsymbol{B}_e^{-1}, \tag{4.24}$$

where $\boldsymbol{T}_e(x)$ is the current triangle basis of element e and $\boldsymbol{B}_e$ is the triangle basis of element e in material space.

4.1.3.2 The Material Point Domain

For the MPM domain, the nodal positions x_{M}^n are the uniform Cartesian grid coordinates at each time step. The grid velocity $\boldsymbol{v}_i^n$ and mass m_i^n are transferred from particles. The nodal movements are conceptual. $\tilde{x}_{\mathrm{M}}^{n+1}$ and $\tilde{v}_{\mathrm{M}}^{n+1}$ will be transferred back to particles for advection.

Similar to FEM, each MPM grid node i is associated with a kernel function $N_i(\boldsymbol{x})$ for the grid to represent the continuous field. Note that the kernel is defined in terms of $\boldsymbol{x}$ rather than $\boldsymbol{X}$ because the grid is essentially a discretization of Ω_M^n – a direct consequence of adopting Updated Lagrangian kinematics. When N_i is evaluated at a particle location $\boldsymbol{x}_q^n$, a shorter notation $N_i(\boldsymbol{x}_q^n) = w_{iq}^n$ is from now on used instead. Here N_i directly takes the current particle location $\boldsymbol{x}_q^n$ as input as opposed to FEM because there is no globally defined reference configuration in MPM and the deformation is evolved over time steps rather than

recomputed using a rest shape. More specifically, the deformation gradient of a particle q is defined as

$$\boldsymbol{F}_q(x) = \sum_i \boldsymbol{x}_i (\nabla w_{ip}^n)^T \boldsymbol{F}_q^n. \tag{4.25}$$

In this paper, without loss of generality, we adopt the quadratic B-spline kernel for $N_i(\boldsymbol{x})$ to avoid MPM's cell-crossing instability (Steffen et al., 2008). Other kernels based on the NURBS (Long et al., 2019), Generalized Interpolation Material Point Method (GIMP) (Bardenhagen and Kober, 2004), Convective Particle Domain Interpolation (CPDI) (Homel et al., 2016; Sadeghirad et al., 2013; Nguyen et al., 2017), or the Dual Domain Material Point (DDMP) (Long et al., 2016; Zhang, 2013) can also be directly used in our framework.

To transfer information between the particles and the grid, we implemented options including the Affine Particle-In-Cell (APIC) method (Jiang et al., 2017b, 2015a) (Table 4.1), Particle-In-Cell (PIC) method (Harlow, 1962) (Table 4.2), and the Fluid-Implicit Particle (FLIP) method (Brackbill et al., 1988) (Table 4.3). Note that other transfer schemes such as XPIC (Hammerquist and Nairn, 2017) can also be applied in our framework in a straightforward manner.

Table 4.1: APIC Particle-Grid Transfer

Particles to grid (APIC)	Grid to particles (APIC)
$m_i^n = \sum_p m_p w_{ip}^n$	$\boldsymbol{v}_p^{n+1} = \sum_i \tilde{\boldsymbol{v}}_i^{n+1} w_{ip}^n$
$\boldsymbol{D}_p^n = \sum_i w_{ip}^n (\boldsymbol{x}_i^n - \boldsymbol{x}_p^n)(\boldsymbol{x}_i^n - \boldsymbol{x}_p^n)^T$	$\boldsymbol{x}_p^{n+1} = \sum_i \tilde{\boldsymbol{x}}_i^{n+1} w_{ip}^n$
$m_i^n \boldsymbol{v}_i^n = \sum_p w_{ip} m_p^n (\boldsymbol{v}_p^n + \boldsymbol{B}_p (\boldsymbol{D}_p)^{-1} (\boldsymbol{x}_i^n - \boldsymbol{x}_p^n))$	$\boldsymbol{B}_p^n = \frac{1}{2} \sum_i w_{ip}^n \Big(\tilde{\boldsymbol{v}}_i^{n+1} (\boldsymbol{x}_i^n - \boldsymbol{x}_p^n + \tilde{\boldsymbol{x}}_i^{n+1} - \boldsymbol{x}_p^{n+1})^T + (\boldsymbol{x}_i^n - \boldsymbol{x}_p^n - \tilde{\boldsymbol{x}}_i^{n+1} + \boldsymbol{x}_p^{n+1})(\tilde{\boldsymbol{v}}_i^{n+1})^T \Big)$
	$\boldsymbol{F}_p^{n+1} = \sum_i \tilde{\boldsymbol{x}}_i^{n+1} (\nabla w_{ip}^n)^T \boldsymbol{F}_p^n$

Table 4.2: PIC Particle-Grid Transfer

Particles to grid (PIC)	Grid to particles (PIC)
$m_i^n = \sum_p m_p w_{ip}^n$ $m_i^n \boldsymbol{v}_i^n = \sum_p w_{ip}^n m_p^n \boldsymbol{v}_p^n$	$\boldsymbol{x}_p^{n+1} = \sum_i \tilde{\boldsymbol{x}}_i^{n+1} w_{ip}^n$ $\boldsymbol{v}_p^{n+1} = \sum_i \tilde{\boldsymbol{v}}_i^{n+1} w_{ip}^n$ $\boldsymbol{F}_p^{n+1} = \sum_i \tilde{\boldsymbol{x}}_i^{n+1} (\nabla w_{ip}^n)^T \boldsymbol{F}_p^n$

Table 4.3: FLIP Particle-Grid Transfer

Particles to grid (FLIP)	Grid to particles (FLIP)
$m_i^n = \sum_p m_p w_{ip}^n$ $m_i^n \boldsymbol{v}_i^n = \sum_p w_{ip}^n m_p^n \boldsymbol{v}_p^n$	$\boldsymbol{x}_p^{n+1} = \sum_i \tilde{\boldsymbol{x}}_i^{n+1} w_{ip}^n$ $\boldsymbol{v}_p^{n+1} = \boldsymbol{v}_p^n + \sum_i w_{ip}^n (\tilde{\boldsymbol{v}}_i^{n+1} - \boldsymbol{v}_i^n)$ $\boldsymbol{F}_p^{n+1} = \sum_i \tilde{\boldsymbol{x}}_i^{n+1} (\nabla w_{ip}^n)^T \boldsymbol{F}_p^n$

4.1.4 The Contact between Domains

4.1.4.1 Contact Potential

Recently for Lagrangian FEM, Li *et al.* (Li et al., 2020, ND) proposed a consistent variational contact model that smoothly approximates the nonsmooth contact phenomena with bounded error, and demonstrated its convergence under refinement for piecewise linear boundary discretization. Here we customize the FEM contact potential (Li et al., ND) to the FEM-MPM coupling setting by defining the inter-surface contact potential between surfaces $\partial\Omega_{\mathrm{M}}$ and $\partial\Omega_{\mathrm{F}}$ to be

$$\int_{\partial\Omega_{\mathrm{M}}^0} b(\min_{\boldsymbol{x}_f \in \partial\Omega_{\mathrm{F}}^t} d^{PP}(\boldsymbol{x}_m, \boldsymbol{x}_f), \hat{d}) \mathbf{d}\boldsymbol{X}_m, \tag{4.26}$$

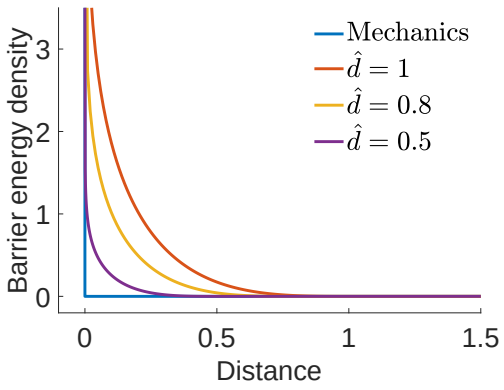


Figure 4.2: **The barrier energy density function Equation (4.27) plotted with different $\hat{d}$.** Decreasing $\hat{d}$ asymptotically matches the discontinuous definition of the contact condition.

where $d^{PP}(\boldsymbol{x}_m, \boldsymbol{x}_f) = \|\boldsymbol{x}_m - \boldsymbol{x}_f\|$ is the point-point distance function, and

$$b(d, \hat{d}) = \begin{cases} -\kappa(\frac{d}{\hat{d}} - 1)^2 \ln(\frac{d}{\hat{d}}) & 0 < d < \hat{d} \\ 0 & d \geq \hat{d} \end{cases} \tag{4.27}$$

is a smoothly clamped barrier function that serves as the contact energy density with d the input distance, $\hat{d}$ a small distance threshold below which contact activates, and κ in Pa the barrier stiffness (Li et al., 2020, ND), which scales the magnitude of contact forces at a certain distance.

Intuitively, $\min_{\boldsymbol{x}_f \in \partial\Omega^t_{\mathrm{F}}} d^{PP}(\boldsymbol{x}_m, \boldsymbol{x}_f)$ is the distance between a material point $\boldsymbol{x}_m \in \partial\Omega^t_{\mathrm{M}}$ and surface $\partial\Omega^t_{\mathrm{F}}$, and Equation (4.26) can be viewed as an integration of the point($\boldsymbol{x}_m$)-surface($\partial\Omega^t_{\mathrm{F}}$) contact energy density over surface $\partial\Omega^0_{\mathrm{M}}$. The barrier function b smoothly increases from 0 to infinity as the input distance decreases from $\hat{d}$ to 0, providing arbitrarily large repulsion to ensure no interpenetration and at the same time bound the contact gap error within $\hat{d}$ (Figure 4.2). As $\hat{d} \to 0$, the approximation error between the contact energy density function b and the real contact phenomenon described in Equation (4.20) decreases, which also makes $\partial\Omega_{\mathrm{M}}$ and $\partial\Omega_{\mathrm{F}}$ interchangeable in the limit. Note that the integration is performed in the material space while the distance is evaluated in the world space.

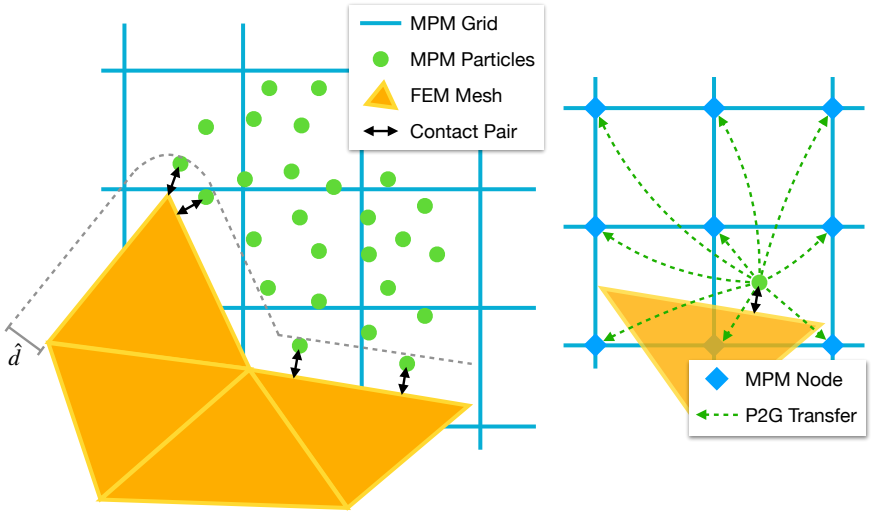


Figure 4.3: **Contact constraint pairs.** Left: Contact activates on all pairs of MPM particles and FEM boundary elements with distance below $\hat{d}$. Right: Contact force is transferred from MPM particles to MPM grids via chain rule.

4.1.4.2 Discretization

After applying FEM and MPM discretization schemes, assuming 2D, the contact potential Equation (4.26) is discretized to be

$$B(x) = \sum_{q \in \mathcal{Q}} \omega_q b(\min_{e \in \mathcal{B}} d^{PE}(\boldsymbol{x}_q, e), \hat{d}), \tag{4.28}$$

where $\mathcal{Q}$ is the set of all MPM particles, ω_q is the integration weight (equivalently, the boundary area) of MPM particle q, $\mathcal{B}$ is the set of FEM boundary edges, and $d^{PE}(\boldsymbol{x}_q, e)$ is the point-edge distance between particle $\boldsymbol{x}_q$ and edge e. Note that the MPM grid nodal positions $\tilde{x}_{\mathrm{M}}$ to be solved and the particle positions $\boldsymbol{x}_q$ after advection using $\tilde{x}_{\mathrm{M}}$ are linearly related through the particle-grid transfer kernel (Figure 4.3 right), and so the contact force on the MPM nodal degrees of freedom can be calculated by applying the chain rule:

$$\frac{\partial B}{\partial x_{\mathrm{M}}} = \sum_{q \in \mathcal{Q}} \left(\frac{\partial \boldsymbol{x}_q}{\partial x_{\mathrm{M}}} \right)^T \frac{\partial B}{\partial \boldsymbol{x}_q}, \tag{4.29}$$

where

$$\left(\frac{\partial \boldsymbol{x}_q}{\partial x_M} \right)_{\alpha, id+\beta} = \delta_{\alpha\beta} w_{iq}^n$$

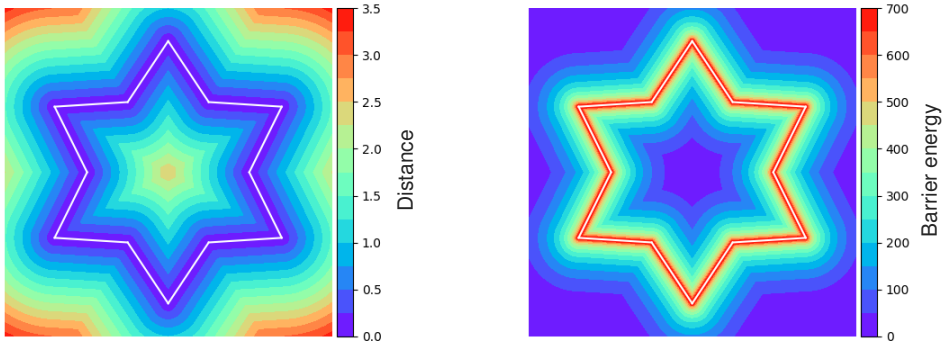


Figure 4.4: A visual demonstration in 2D of (left) the unsigned distance function to a segmented mesh, and (right) the corresponding barrier function (4.27) visualized with an exaggerated $\hat{d}$ parameter.

with $\alpha, \beta = 1, 2, ..., d$ and $d = 2$ or 3 the spatial dimension since $\boldsymbol{x}_q = \sum_i w_{iq}^n \boldsymbol{x}_i$.

Ideally, ω_q should be zero for interior particles. It should reveal the proportional surface area of boundary particles. Since FEM boundary elements are always outside the MPM domain and the barrier function b is only activated at a small distance, the activation of b can be applied to conveniently decide whether an MPM particle is at the boundary or the interior without explicitly identifying the MPM domain boundary in each time step (Figure 4.3 left). Then assuming a close to uniform particle distribution to be maintained throughout the simulation, ω_q can be set to $2\sqrt{V_q^0/\pi}$ in 2D and $\pi \left(3V_q^0/\left(4\pi\right)\right)^{\frac{2}{3}}$ in 3D for all particles, which is the area of the largest cross section of a spherical particle q.

The minimization operator in the potential (Equation (4.28)) helps to compute the point-polyline distance from point-edge distances. As the barrier function b is monotonically decreasing, the potential can be rewritten as

$$B(x) = \sum_{q \in \mathcal{Q}} \omega_q \max_{e \in \mathcal{B}} b(d^{PE}(\boldsymbol{x}_q, e), \hat{d}). \tag{4.30}$$

Due to the existence of a max operator, it is only C^0 continuous, making the incremental potential challenging to be efficiently minimized by gradient-based optimization methods like Newton's method. Since the barrier function b is with local support around each boundary

element, and that it maps a majority of the activate distances to tiny potential values (Figure 4.4), the maximization of the potential field can be well approximated by summation, with the duplicate potential around FEM boundary nodes compensated by subtraction as proposed by Li *et al.* (Li et al., ND):

$$B(x) = \sum_{q \in \mathcal{Q}} \omega_q \left(\sum_{e \in \mathcal{B}} b(d^{PE}(\boldsymbol{x}_q, e), \hat{d}) - \sum_{k \in \hat{\mathcal{B}}} (\eta_k - 1)\, b(d^{PP}(\boldsymbol{x}_q, \boldsymbol{x}_k), \hat{d}) \right), \tag{4.31}$$

where $\hat{\mathcal{B}}$ is the set of all FEM boundary nodes and η_k is the number of FEM boundary edges incident to node k. For closed manifold domains in 2D, $\eta_k = 2$ for all k.

Similarly, in 3D, the discretized contact potential becomes

$$B(x) = \sum_{q \in \mathcal{Q}} \omega_q \left(\sum_{t \in \mathcal{B}} b(d^{PT}(\boldsymbol{x}_q, t), \hat{d}) - \sum_{e \in \hat{\mathcal{B}}} (\eta_e - 1)\, b(d^{PE}(\boldsymbol{x}_q, e), \hat{d}) + \sum_{\boldsymbol{x}_p \in \tilde{\mathcal{B}}} b(d^{PP}(\boldsymbol{x}_q, \boldsymbol{x}_p), \hat{d}) \right), \tag{4.32}$$

where $\mathcal{B}$ is now the set of all FEM boundary triangles, $\hat{\mathcal{B}}$ is the set of all edges on the FEM boundary with η_e the number of FEM boundary triangles incident to edge e, $\tilde{\mathcal{B}}$ the set of all nodes that are in the interior of the FEM boundary surface mesh, and $d^{PT}(\boldsymbol{x}_q, t)$ the point-triangle distance between particle $\boldsymbol{x}_q$ and triangle t. For closed manifold domains in 3D, $\eta_e = 2$ for all e.

Adding the contact potential into the incremental potential, the minimization problem for time integration is now fully unconstrained:

$$\min_x : \frac{1}{2}\|x - \hat{x}^n\|_M^2 + 2\alpha\beta\Delta t^2 \left(\Psi(x) + B(x)\right). \tag{4.33}$$

Since the distance values measured for contacting MPM particle - FEM simplex pairs are all unsigned, Problem (4.33) may contain a local optimum at configurations with intersections. To be consistent with the continuous constraint Equation (4.20), it is also constrained that the iterates always stay in the feasible region on one side of the barrier without crossing. This is achieved by applying the interior-point filter line-search algorithm (Wächter and Biegler,

2006) with continuous collision detection (CCD) (Brochu et al., 2012; Li et al., 2021a).

4.1.4.3 Friction

To model frictional contact, local frictional forces F_k can be added for every active contact pair k. For each such pair k, at the current state x, a consistently oriented sliding basis $T_k(x) \in \mathbb{R}^{dm\times(d-1)}$ can be constructed, where m is the total number of colliding nodes and d is the dimension of space, such that $\boldsymbol{u}_k = T_k(x)^T(\Delta t v^n + \Delta t^2((1-\gamma)a^n + \gamma a^{n+1})) \in \mathbb{R}^{d-1}$ provides the local relative sliding displacement in the frame orthogonal to the distance gradient. The corresponding sliding velocity is then $\boldsymbol{v}_k = \boldsymbol{u}_k/\Delta t \in \mathbb{R}^{d-1}$.

Maximizing dissipation rate subject to the Coulomb constraint defines friction forces variationally (Moreau, 2011; Goyal et al., 1991)

$$F_k(x) = T_k(x) \operatorname*{arg\,min}_{\boldsymbol{\beta}\in\mathbb{R}^{d-1}} \boldsymbol{\beta}^T \boldsymbol{v}_k \quad \text{s.t.} \quad \|\boldsymbol{\beta}\| \le \mu\lambda_k, \tag{4.34}$$

where λ_k is the contact force magnitude and μ is the local friction coefficient. This is equivalent to

$$F_k(x) = -\mu\lambda_k T_k(x) f(\|\boldsymbol{u}_k\|)\, \mathbf{s}(\boldsymbol{u}_k), \tag{4.35}$$

with $\mathbf{s}(\boldsymbol{u}_k) = \frac{\boldsymbol{u}_k}{\|\boldsymbol{u}_k\|}$ when $\|\boldsymbol{u}_k\| > 0$, while $\mathbf{s}(\boldsymbol{u}_k)$ takes any unit vector when $\|\boldsymbol{u}_k\| = 0$. The friction magnitude function, f, is nonsmooth with respect to $\boldsymbol{u}_k$ since $f(\|\boldsymbol{u}_k\|) = 1$ when $\|\boldsymbol{u}_k\| > 0$, and $f(\|\boldsymbol{u}_k\|) \in [0,1]$ when $\|\boldsymbol{u}_k\| = 0$. This nonsmoothness would severely slow and even break convergence of gradient-based optimization.

To enable efficient and stable optimization, the friction-velocity relation in the transition to static friction can be mollified by replacing f with a smoothly approximated function. Following Li *et al.* (Li et al., 2020), we use

$$f_1(y) = \begin{cases} -\frac{y^2}{\epsilon_v^2\Delta t^2} + \frac{2y}{\epsilon_v \Delta t}, & y \in [0, \Delta t\epsilon_v) \\ 1, & y \ge \Delta t\epsilon_v, \end{cases} \tag{4.36}$$

where $f_1'(\Delta t\epsilon_v) = 0$ and a velocity magnitude bound ϵ_v (in units of m/s) below which sliding

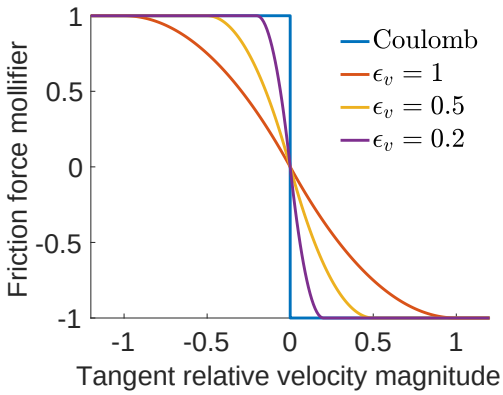


Figure 4.5: **Friction mollifier plotted with different ϵ_v.** Decreasing ϵ_v asymptotically matches the discontinuous Coulomb friction model.

velocities $\boldsymbol{v}_k$ are treated as static is defined for bounded approximation error (Figure 4.5).

Note that the velocity used in our friction model on the MPM side is the interpolated grid velocity at particle quadrature locations, rather than the particle velocity after grid-to-particle transfer. This makes the velocity seen by frictional forces independent from the choice of the particle-grid transfer scheme. This is important because for example in FLIP, the particle velocity does not reflect its displacement ($\boldsymbol{v}_q^{n+1} \neq (\boldsymbol{x}_p^{n+1} - \boldsymbol{x}_p^n)/\Delta t$) and thus should not be used to define friction in an implicit solve.

However, challenges remain on incorporating friction into the optimization time integration. A major problem is that friction is not a conservative force and there is no well-defined potential such that taking the opposite of its gradient produces the frictional force. Therefore, following Li *et al.* (Li et al., 2020), we fix the friction constraint set $\mathcal{F}$ along with the normal force magnitude λ and the tangent operator T during the nonlinear optimization to the last updated value $\mathcal{F}^j = \mathcal{F}(x^j)$, $\lambda^j = \lambda(x^j)$, and $T^j = T(x^j)$, which then makes the lagged friction force integrable with the pseudo-potential

$$D(x) = \sum_{k \in \mathcal{F}^j} \mu \lambda_k^j f_0(\|\bar{\boldsymbol{u}}_k\|), \tag{4.37}$$

where $\mathcal{F}^j$ is the set of all contact pairs with nonzero λ_k^j, $f_0'(y) = f_1(y)$, $\bar{\boldsymbol{u}}_k = (T_k^j)^T(\Delta t v^n + \Delta t^2((1-\gamma)a^n + \gamma a^{n+1}))$ and so we have $-\nabla D(x) = -\sum_{k \in \mathcal{F}^j} \mu \lambda_k^j T_k^j f_1(\|\bar{\boldsymbol{u}}_k\|)\, \mathbf{s}(\bar{\boldsymbol{u}}_k)$, which is a semi-implicit discretization of the frictional force with lagged variables λ_k^j and T_k^j. Then we

can iteratively alternate between the nonlinear optimization with fixed $\mathcal{F}$, λ, and T given as

$$\min_x : E(x) = \frac{1}{2}\|x - \hat{x}^n\|_M^2 + 2\alpha\beta\Delta t^2 \left(\Psi(x) + B(x) + D(x)\right), \tag{4.38}$$

and friction update until convergence (Algorithm 2). Although the friction convergence is not guaranteed for arbitrarily large time step sizes due to the nonlinearity and asymmetry of the problem, we have confirmed that all our experiments converge with the practical time step sizes applied (Section 4.1.6).

4.1.4.4 Irregular Boundaries for MPM

In the BFEMP framework, an experimental setup with a subset or all of the FEM nodes prescribed with Dirichlet boundary conditions on their displacements can be applied to model irregular boundaries for MPM. This can not only resolve detailed boundary geometries even when the MPM grid is relatively coarse (Section 4.1.6.2), but also provide accurate and controllable friction on the boundary (Section 4.1.6.4).

4.1.5 Nonlinear Optimization

The time integration framework of BFEMP for one time step is outlined in Algorithm 2. MPM particle-grid transfers are performed in the beginning (line 2) and the end (line 12). On the MPM grid and the FEM mesh, the minimization of incremental potential with lagged friction (line 7) is alternated with the friction update (line 9) until convergence to the fully implicit friction solution.

Applying the projected Newton's method (Teran et al., 2005) for incremental potential minimization (Algorithm 3), we compute the proxy matrix H by projecting the local Hessian of every elasticity, barrier, and friction stencil to its closest positive semi-definite form by zeroing out the negative eigenvalues, and then summing them up together with the mass matrix M^n (line 5). The search direction p is computed by factorizing H and back-solving it on $-\nabla E(x)$ using CHOLMOD (Chen et al., 2008) (line 6). To obtain global convergence,

Algorithm 2 BFEMP Time Integration

procedure TIMEINTEGRATION(x_{F}^n, v_{F}^n, M_{F}, x_{P}^n, v_{P}^n, M_{P}, Δt) ▷ subscript P is for stacked particle variables
  x_{M}^n, v_{M}^n, $M_{\mathrm{M}}^n \leftarrow$ particleToGrid(x_{P}^n, v_{P}^n, M_{P}) ▷ Table 4.1, 4.2, and 4.3
  $\tilde{x}_{\mathrm{F}}^{n+1} \leftarrow x_{\mathrm{F}}^n$, $\tilde{x}_{\mathrm{M}}^{n+1} \leftarrow x_{\mathrm{M}}^n$ ▷ for initial guess
  $j \leftarrow 0$
  $\mathcal{F}^j$, λ^j, $T^j \leftarrow$ computeFrictionOperator($\tilde{x}_{\mathrm{F}}^{n+1}$, $\tilde{x}_{\mathrm{M}}^{n+1}$) ▷ Section 4.1.4.3
  do
    $\begin{bmatrix}\tilde{x}_{\mathrm{F}}^{n+1}\\ \tilde{x}_{\mathrm{M}}^{n+1}\end{bmatrix}, \begin{bmatrix}\tilde{v}_{\mathrm{F}}^{n+1}\\ \tilde{v}_{\mathrm{M}}^{n+1}\end{bmatrix} \leftarrow$ MinimizeIP($\begin{bmatrix}x_{\mathrm{F}}^n\\ x_{\mathrm{M}}^n\end{bmatrix}, \begin{bmatrix}v_{\mathrm{F}}^n\\ v_{\mathrm{M}}^n\end{bmatrix}, \begin{bmatrix}M_{\mathrm{F}} & \\ & M_{\mathrm{M}}^n\end{bmatrix}, \Delta t, \mathcal{F}^j, \lambda^j, T^j, \begin{bmatrix}\tilde{x}_{\mathrm{F}}^{n+1}\\ \tilde{x}_{\mathrm{M}}^{n+1}\end{bmatrix}$) ▷ Algorithm 3
    $j \leftarrow j + 1$
    $\mathcal{F}^j$, λ^j, $T^j \leftarrow$ computeFrictionOperator($\tilde{x}_{\mathrm{F}}^{n+1}$, $\tilde{x}_{\mathrm{M}}^{n+1}$) ▷ Section 4.1.4.3
  while friction not converged ▷ Section 4.1.5
  $x_{\mathrm{F}}^{n+1} \leftarrow \tilde{x}_{\mathrm{F}}^{n+1}$, $v_{\mathrm{F}}^{n+1} \leftarrow \tilde{v}_{\mathrm{F}}^{n+1}$
  x_{P}^{n+1}, $v_{\mathrm{P}}^{n+1} \leftarrow$ gridToParticle($\tilde{x}_{\mathrm{M}}^{n+1}$, $\tilde{v}_{\mathrm{M}}^{n+1}$) ▷ Table 4.1, 4.2, and 4.3
  return x_{F}^{n+1}, v_{F}^{n+1}, x_{P}^{n+1}, v_{P}^{n+1}
end procedure

Algorithm 3 Line Search Method for Incremental Potential Minimization

1: **procedure** MINIMIZEIP(x^n, v^n, M^n, Δt, $\mathcal{F}^j$, λ^j, T^j, $\bar{x}$)
2: $x \leftarrow \bar{x}$ ▷ initial guess
3: $E_{\mathrm{prev}} \leftarrow E(x)$, $x_{\mathrm{prev}} \leftarrow x$ ▷ $E(x)$ defined in (4.38) also depends on x^n, v^n, M^n, Δt, $\mathcal{F}^j$, λ^j, T^j
4: **do**
5: $H \leftarrow$ computeProxyMatrix(x) ▷ applying projected Newton (Teran et al., 2005)
6: $p \leftarrow -H^{-1}\nabla E(x)$ ▷ solved using CHOLMOD (Chen et al., 2008)
7: $\tau \leftarrow$ initStepSize(x) ▷ line search filtering (Wächter and Biegler, 2006)
8: **do** ▷ Armijo line search (Nocedal and Wright, 1999)
9: $x \leftarrow x_{\mathrm{prev}} + \tau p$
10: $\tau \leftarrow \tau/2$
11: **while** $E(x) > E_{\mathrm{prev}}$
12: $E_{\mathrm{prev}} \leftarrow E(x)$, $x_{\mathrm{prev}} \leftarrow x$
13: **while** $\|p\|_\infty/\Delta t > \epsilon_d$ ▷ Section 4.1.5
14: $\tilde{x}^{n+1} \leftarrow x$, $\tilde{v}^{n+1} \leftarrow v^n + \frac{1}{\Delta t}(M^n)^{-1}((\gamma-1)\nabla E(x^n) - \gamma\nabla E(x))$ ▷ Section 4.1.3
15: **return** $\tilde{x}^{n+1}$, $\tilde{v}^{n+1}$
16: **end procedure**

the backtracking line search that ensures the decrease of energy is applied (line 8 to 11), starting from a large feasible step size that avoids interpenetration and deformation gradient degeneracy (line 7). After converging to a local optimum, velocity is updated with the newly obtained position (line 14) and returned together (line 15). Here we use the infinity norm

of the Newton increment (search direction p) in the unit of velocity (m/s) for the stopping criteria, which provides a 2nd-order approximation on the distance to the true solution. Similarly, friction convergence in Algorithm 2 is also determined this way, but with $\mathcal{F}$, λ, and T computed using the current x.

Along Newton's search direction p, we compute the largest step size that will first result in a 0 distance on any contact pair or a 0 determinant on any deformation gradient. We then set the initial line search step size to be $0.9\times$ of this critical value. The critical value for 0 distance is computed via continuous collision detection (CCD) (Li et al., 2021a), and for 0 determinant it is just the smallest positive real root of a polynomial equation (Li et al., 2021e). This ensures that interpenetration or deformation gradient degeneracy could never happen throughout the simulation since the following backtracks always result in step sizes smaller than the critical value.

The numerical parameters in BFEMP all have physical meanings and directly control the extent of approximation to the continuous problem. To summarize, we have $\hat{d}$ (contact activation distance in m), ϵ_v (stick-slip velocity threshold in m/s), ϵ_d (Newton tolerance in m/s), and physical parameter κ (barrier stiffness in Pa). Here κ also affects the convergence speed of the projected Newton method (Algorithm 3), but the convergence is always guaranteed eventually. In our experiments, we observed that setting κ several orders of magnitude smaller than the average elasticity stiffness of the objects in the simulation can provide efficient convergence.

4.1.6 Numerical Simulations

In this section, we provide 6 examples in 2D and 1 example in 3D to verify the contact model and the friction model in the proposed BFEMP approach. The numerical parameters $\hat{d}$, ϵ_v , ϵ_d, and physical parameter κ are all reported respectively in each experiment. If not mentioned otherwise, all elasticities are with the neo-Hookean model, and all particle-grid transfer schemes are with APIC. The visualized stresses are all the von Mises stress.

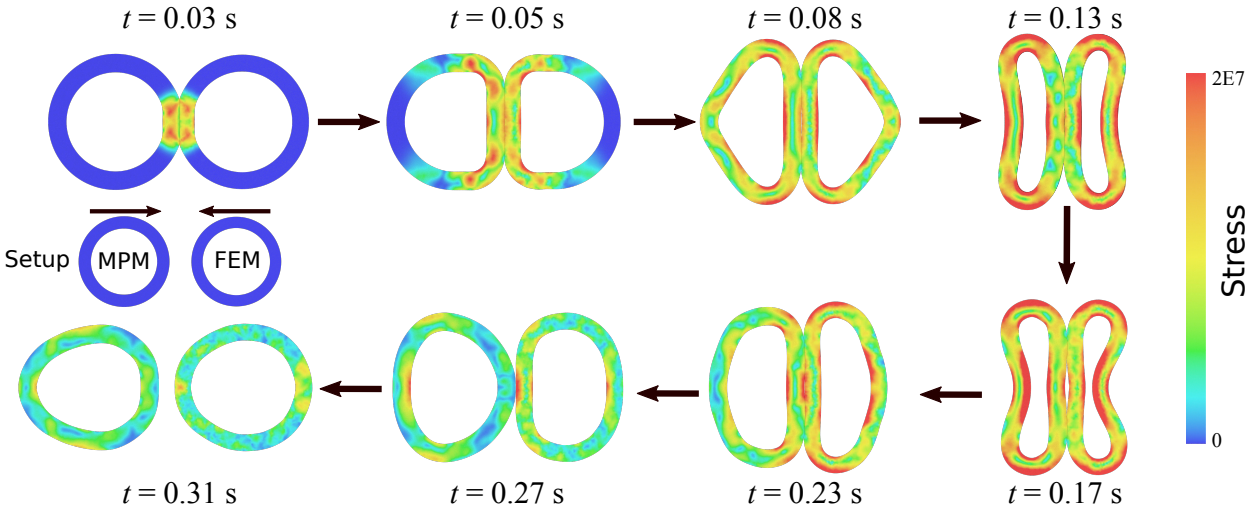


Figure 4.6: **Colliding rings.** The experiment setup and stress wave propagation over time.

4.1.6.1 Momentum and Energy Study

The collision between two elastic rings is simulated to verify the momentum and energy behavior of BFEMP and to demonstrate the robustness of our framework in handling large deformation. This example is modified from the MPM-MPM contact version in (Huang et al., 2010).

The experiment setup is shown in the left-middle subfigure of Figure 4.6. The two rings are identical except that the left ring is discretized with MPM, and the right one is discretized with FEM. The inner radius of the ring shape is $3m$, and the outer radius of the ring shape is $4m$. The Young's modulus is $E = 10^8 Pa$, the Poisson's ratio is $\nu = 0.2$, and the density is $\rho = 1000 kg/m^2$. The MPM ring is discretized by 20098 particles, where the grid spacing is $0.1m$. The FEM ring is discretized by 1830 vertices and 3310 triangles. The gravitational force and the frictional forces are not included. The two rings are placed $2m$ apart and then move towards each other with an initial speed of $40m/s$. The contact active distance and the contact stiffness are set to $\hat{d} = 10^{-2}m$ and $\kappa = 10^7 Pa$ respectively. To minimize numerical dissipation, we use the Newmark time integrator with time step size $\Delta t = 2 \times 10^{-4}s$. The Newton tolerance is $\epsilon_d = 10^{-6}m/s$. We also compare the APIC and FLIP transfer schemes. Their differences in displacement and stress are small, so only the stress wave propagation with APIC is shown in Figure 4.6. The energies and momenta over time are plotted for both

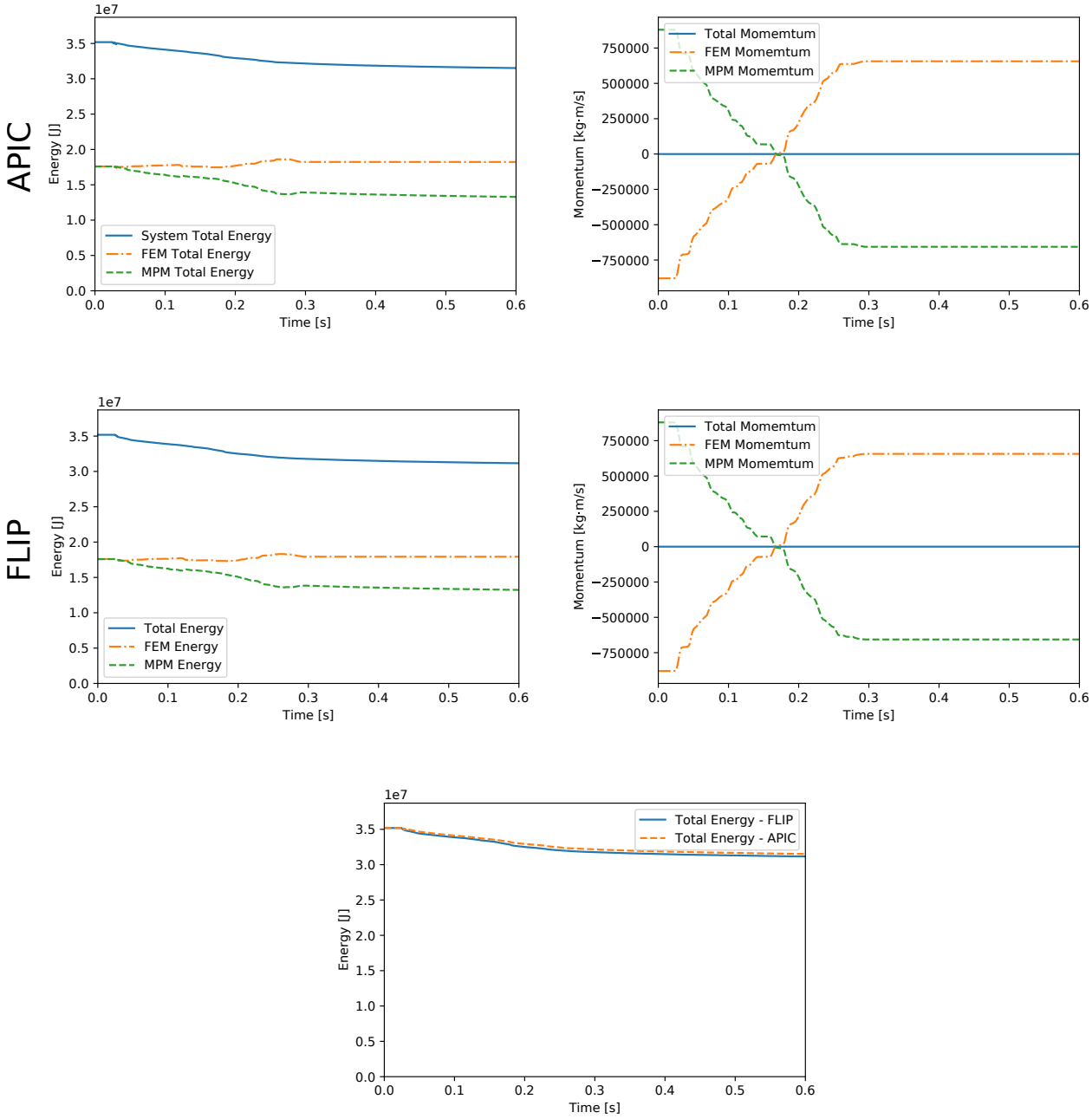


Figure 4.7: **Momentum and Energy Behavior.** The energy and momentum plot for APIC and FLIP transfer schemes.

APIC and FLIP in Figure 4.7.

The collision happens between $0.025s$ and $0.293s$. The symmetry of stress patterns is preserved during the collision. The system's total momentum is perfectly preserved with both transfer schemes. Part of the energy is lost during the collision: 8.57% energy is lost with APIC, and 9.67% with FLIP. After the rings are separated, the FEM ring preserves its

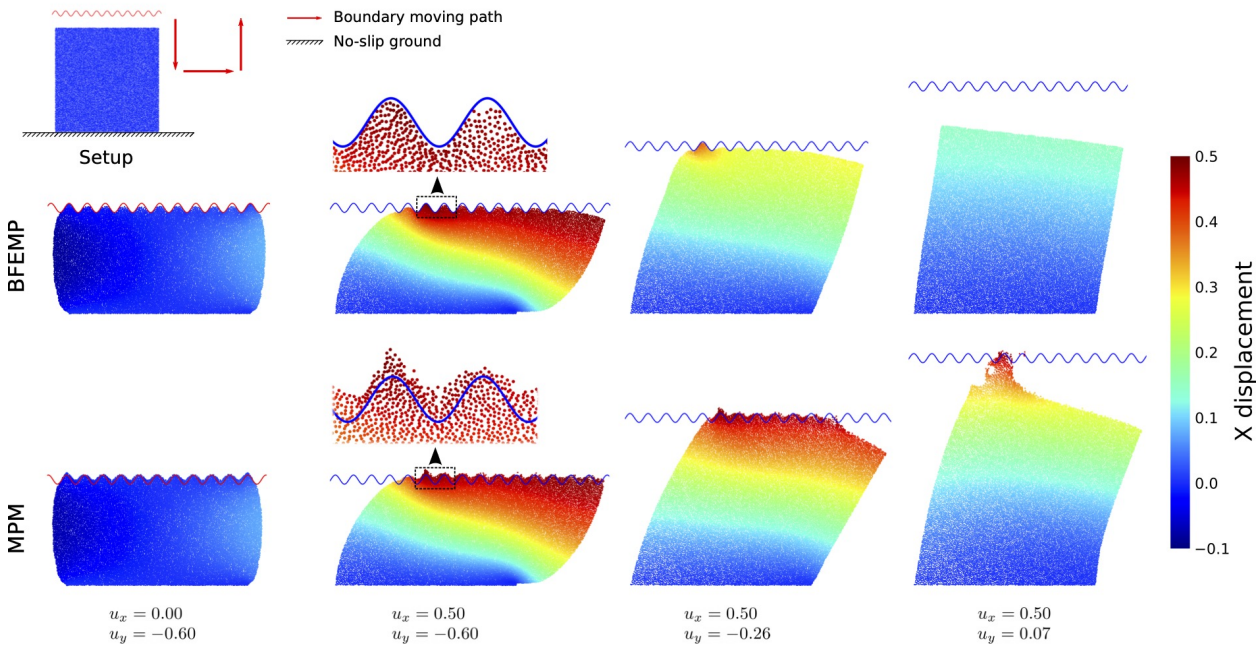


Figure 4.8: **FEM as Boundary Condition.** The friction-free interaction between a sine wave shape boundary and an MPM cube are simulated to compare BFEMP based slip boundary condition and traditional level set based slip boundary condition. (u_x, u_y) is the displacement of the sine wave w.r.t. its initial position. BFEMP based slip boundary condition can guarantee non-penetration and doesn't have adhesive forces when it is separating from the object.

energy over time, while the MPM ring gradually loses energy, primarily due to numerical dissipation in the particle-grid transfers.

4.1.6.2 FEM as Contact Boundary for MPM

The guaranteed impenetrability between MPM particles and FEM boundaries makes BFEMP a natural strategy for enforcing kinematic separable boundary conditions in MPM simulations. Here we test the friction-free interaction between a sine wave shape boundary and an MPM cube. The BFEMP-based boundary condition is compared with a level-set based slip boundary condition, which enforces a zero normal relative velocity condition at each grid node inside the sine wave's level set, i.e., at each time step, for those nodes that are within the level set, their normal velocities along the level set interface are prescribed, so that the original unconstrained optimization 4.22 for the time integration are solved with these equality constraints.

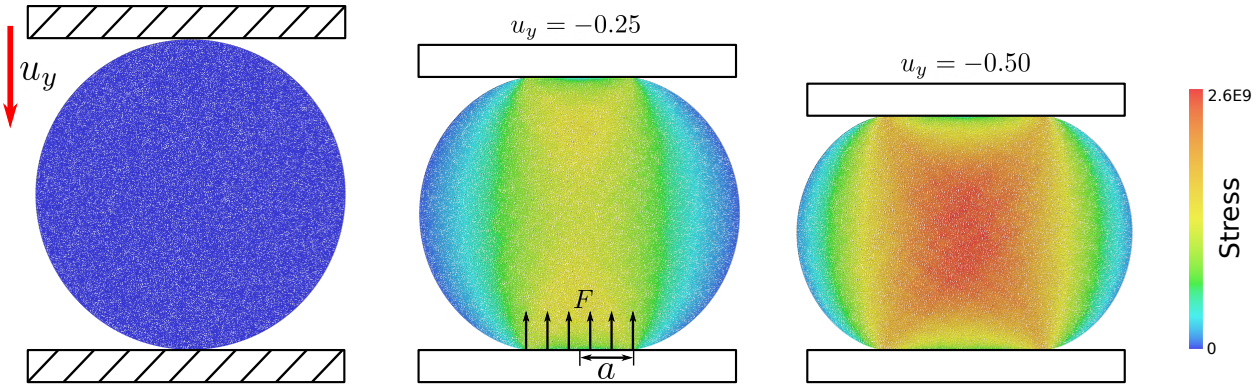


Figure 4.9: **Brazilian Disk Test.** The experiment setup and the compression procedure are shown here. The contact force and contact radius are illustrated in the middle figure.

The experiment setup is shown in the left-top subfigure of Figure 4.8. A $1m \times 1m$ elastic box with Young's modulus $E = 10^6 Pa$, Poisson's ratio $\nu = 0.2$ and $\rho = 1000 kg/m^2$ is placed on a no-slip ground. It is discretized by 21026 particles, with grid spacing $0.02m$. A sine wave boundary is placed $0.2m$ above the box, whose contour is determined by $y = \frac{1}{40} \cos \frac{2\pi}{0.1} x$. For BFEMP, the sine wave boundary condition is discretized by a FEM mesh with prescribed displacements at each time step. While for MPM, it is described by an analytical level set. The sine wave boundary first moves $0.6m$ downwards, then $0.5m$ to the left, and finally upwards until separation. The moving speed is $1m/s$ all the way. The contact active distance and the contact stiffness is set to $\hat{d} = 10^{-3} m$ and $\kappa = 10^4 Pa$ respectively. The implicit Euler time integration with time step $h = 10^{-3} s$ is used. The Newton tolerance is set to $\epsilon_d = 10^{-4} m/s$.

As shown in Figure 4.8, the BFEMP-based boundary condition more accurately resolves the complex boundary geometry without exhibiting any numerical adhesive forces when the boundary is separating from the cube. With the level-set based slip condition, particles will penetrate the boundary because the boundary condition is only defined on the MPM grid in a "smeared out" manner. The numerical adhesive force comes from that at each time step, the grid with slip condition is locked within some plane. On the contrary, with BFEMP, MPM particles can freely move around outside the FEM mesh.

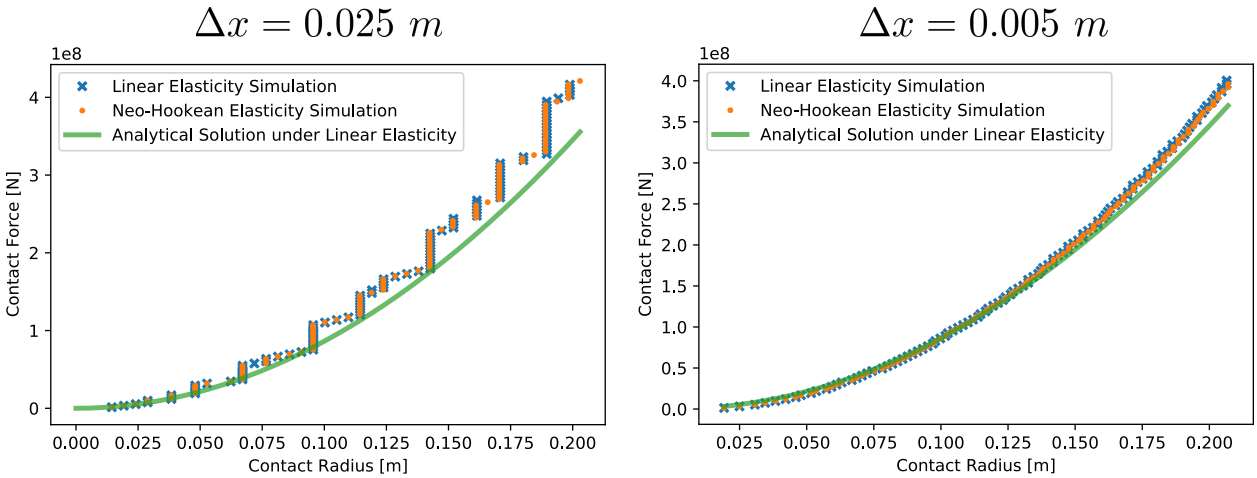


Figure 4.10: **Brazilian Disk Test.** Within the small deformation range, our contact model fits well with Hertzian contact theory. The non-smoothness of the measured radius from the simulation results can be alleviated as the resolution increases.

4.1.6.3 Brazilian Disk Test

To verify the accuracy of the contact model, BFEMP is studied on the Brazilian disk test, which is a special case of the plane Hertzian contact problem (Barber, 2009; Liu and Sun, 2020). The Brazilian disk test can be used for tensile strength testing, which involves a 2D elastic disk squeezed between two rigid objects. We use a fixed rigid plate and a moving rigid plate to simulate the compression procedure. According to the Hertzian contact model, the contact force F and the contact radius a have the following relation:

$$F = \frac{\pi}{4} \frac{E}{1-\nu^2} \frac{a^2}{R}. \tag{4.39}$$

The contact force and the contact radius are illustrated in Figure 4.10.

In this experiment, the radius of the MPM disk is $1m$. It is composed of 42920 particles with MPM grid spacing $\Delta x = 0.025m$. The Young's modulus is $E = 10^{10}Pa$, and the Poisson's ratio is $\nu = 0.3$. To reduce the inertial effect, we artificially decrease the density of the material, which is set to $\rho = 100kg/m^2$. The contact active distance is $\hat{d} = 10^{-4}m$ and the contact stiffness is $\kappa = 10^4 Pa$. The two plates are discretized with FEM. Each of them is composed of four vertices and two triangles. The fixed plate is placed $\hat{d}$ below the disk, and the moving plate is placed $\hat{d}$ above the disk. The constant velocity $0.1m/s$ of the

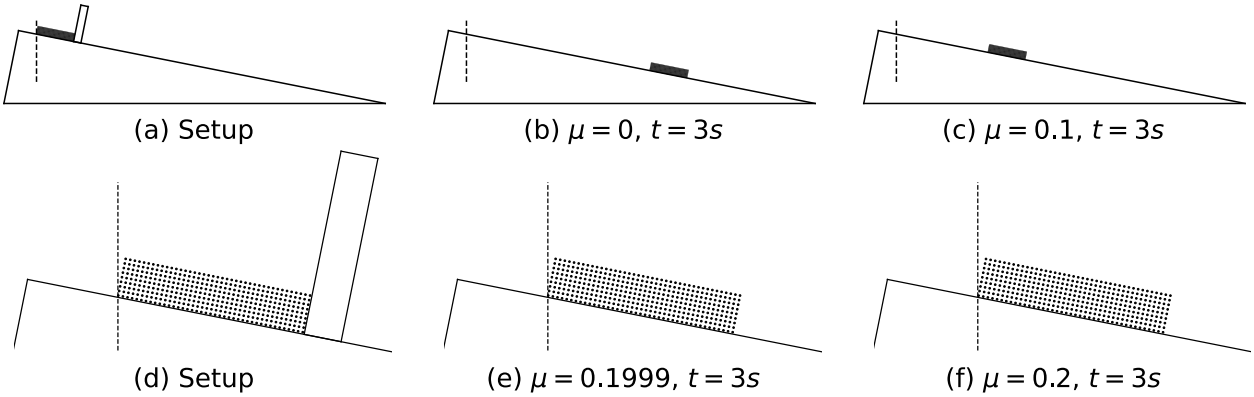


Figure 4.11: **Critical Value of Friction Coefficient.** (a,d) initial configuration with the extra support; (b,c,e) results at $t = 3s$ of $\mu = 0$, 0.1, and 0.1999, sliding distances all matching analytical solutions; (f) the result at $t = 3s$ of $\mu = 0.2$, static solution with sliding error bounded by ϵ_v.

moving plate is enforced by prescribing its displacements at each time step. The simulation is performed with implicit Euler time integration with time step size $h = 10^{-2}s$ and the Newton tolerance is $\epsilon_d = 10^{-8}m/s$. Friction coefficient $\mu = 1$ is used to prevent the disk from slipping.

The Hertzian model requires to measure the contact radius a. Following (Liu and Sun, 2020), we use half of the horizontal range of the particles within the contact distance around the bottom FEM plate to approximate it. Here we test both linear elasticity and neo-Hookean elasticity. The compression procedure in Figure 4.9 is visualized for the linear elasticity case. The (a, F) data points within the small deformation range from the two simulations and the analytical $F - a$ relation from the Hertzian contact model are plotted in Figure 4.10. The non-smoothness of the measured radius from the simulation results is due to the inaccurate approximation of the radius a through a finite number of particles. This non-smoothness can be alleviated as the resolution increases. Despite that, we observe a qualitative match between the simulated data and the Hertzian contact theory.

4.1.6.4 Critical Value of Friction Coefficient

To verify the accuracy of BFEMP's friction model, an experiment with a stiff MPM box resting or sliding on a fixed FEM slope (or BFEMP's friction-controllable boundary condition)

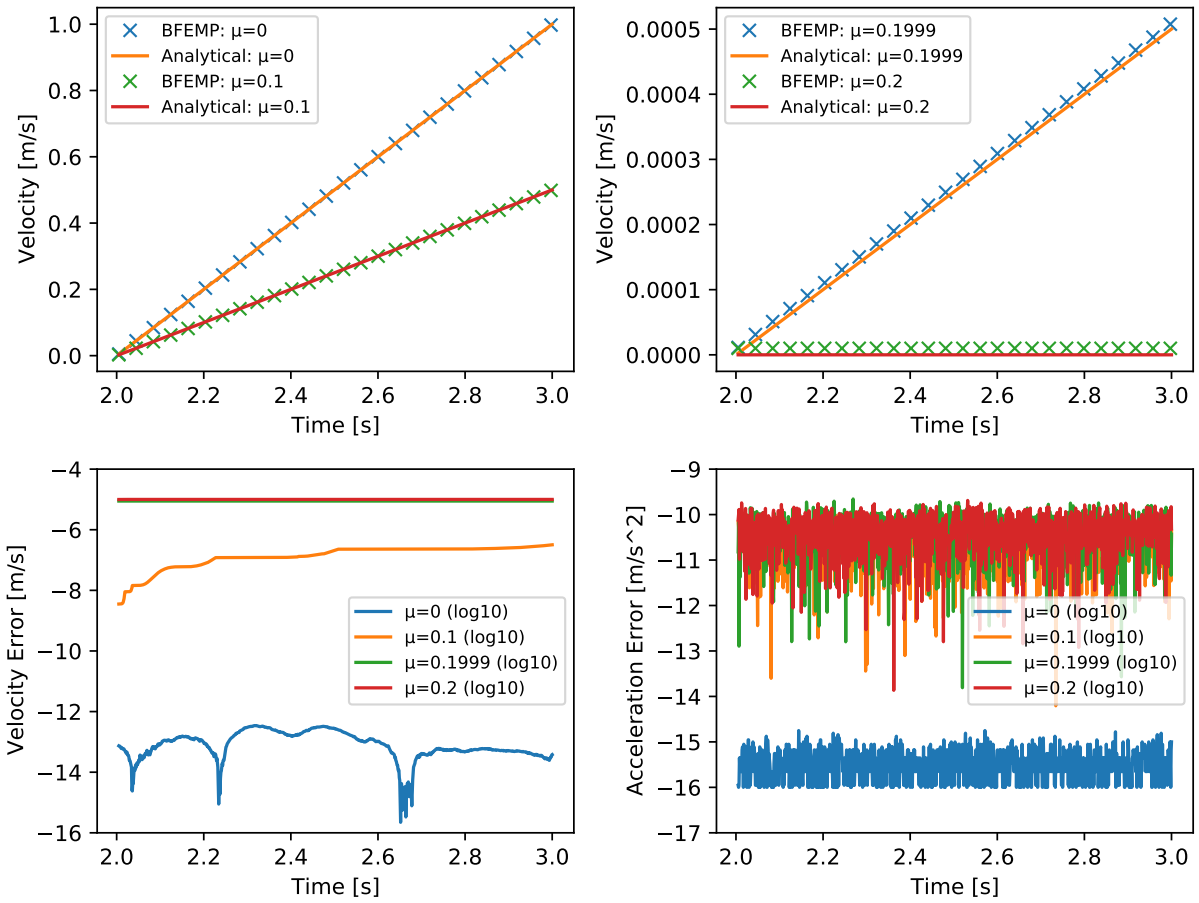


Figure 4.12: **Critical Value of Friction Coefficient.** At all friction coefficients, including $\mu = 0$ (no friction), $\mu = 0.1$, $\mu = 0.1999$ (99.95% of the critical value), and $\mu = 0.2$ (the critical value), the velocities and accelerations over the releasing period ($2s$ to $3s$) are all accurately matching the analytical solutions.

with a certain friction coefficient is created. When a rigid box is placed on a slope with zero initial velocity, its acceleration has the following analytical form:

$$a_x = \begin{cases} g(\sin\theta - \mu\cos\theta), & \theta \geq \tan\theta, \\ 0, & \theta < \tan\theta, \end{cases} \tag{4.40}$$

where μ is the friction coefficient between the box and the slope, g is the gravity acceleration, $\theta \in [0, \pi/4)$ is the inclined angle of the slope. Experiments show that BFEMP's friction model matches analytical solutions on sliding dynamics and critical value of friction coefficient both with bounded and small approximation error.

The initial configuration of this example is obtained by placing the MPM box $\hat{d}$ away from the slope, placing another fixed plane perpendicular to the slope on the side of the box where it may slide (also $\hat{d}$ away), and then simulate under gravity ($g = 5.10m/s^2$) without friction until the box becomes static (Figure 4.11a). After obtaining the initial configuration, the slope test simulation is performed without the extra plane and with multiple different friction coefficients for each test (Figure 4.11b,c,d).

Here the MPM box is $0.1m \times 0.02m$, composed of 369 particles (grid $dx = 0.005$) with density $\rho = 100kg/m^2$, Young's modulus $E = 4.0 \times 10^{12}Pa$ and Poisson's ratio $\nu = 0.2$. Slopes with friction coefficient $\mu = 0$, 0.1, 0.1999, and 0.2 have been tested, all with contact active distance $\hat{d} = 0.001m$, contact stiffness $\kappa = 10^6Pa$, static friction velocity threshold $\epsilon_v = 10^{-5}m/s$, and with the lagged normal forces in friction iteratively updated until converging to a solution with fully-implicit friction. All simulations are using implicit Euler time integration with time step size $h = 0.001s$, and the Newton tolerance is set to $\epsilon_d = 10^{-8}m/s$.

With sliding velocity and acceleration of the box's center of mass plotted over time (Figure 4.12), they have all been shown to well match analytical solutions within 0.01% relative errors. Even for $\mu = 0.1999$ (99.95% that of the critical coefficient), the sliding behavior can still be accurately captured. For $\mu = 0.2$, it is also confirmed that the acceleration vanishes, and the velocity throughout the simulation is around ϵ_v, the static friction velocity threshold in BFEMP's approximation to provide the static friction force in the same magnitude as dynamic friction.

4.1.6.5 Convergence under Refinement

To verify the convergence under refinement property of BFEMP, an example with a soft MPM box stacking on a soft FEM box is created. A series of experiments with increasing resolutions and decreasing contact active distances are simulated to study the convergence rate under refinement. Results show that BFEMP can achieve a second-order convergence rate.

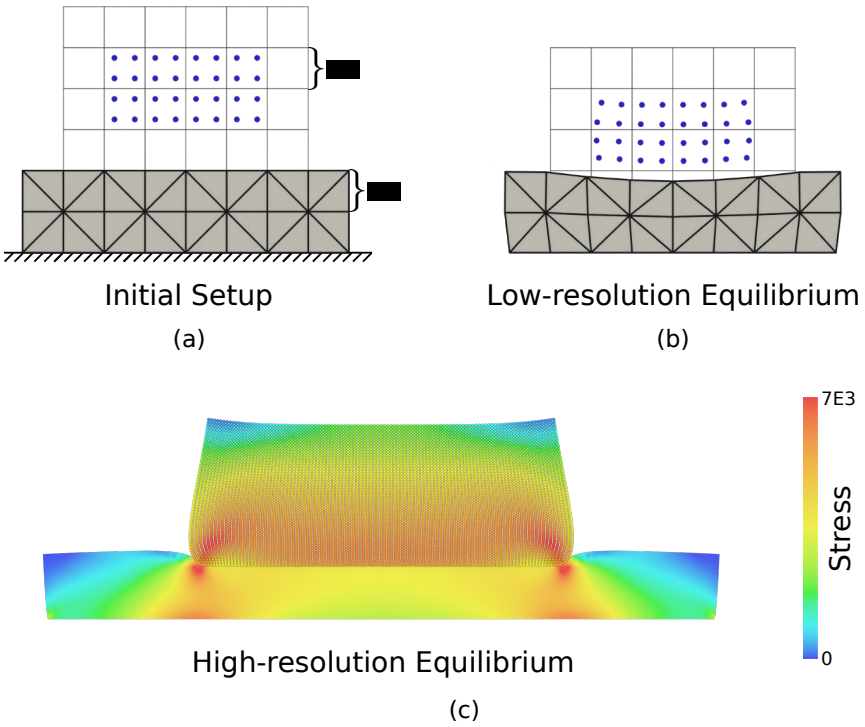


Figure 4.13: **Convergence under Refinement.** (a) Experiment setup. (b) The final equilibrium under low resolution. (c) The final equilibrium under high resolution. Stress pattern is visualized.

The initial configuration is illustrated in Figure 4.13 (a). The MPM box is with size $2m \times 1m$, Young's modulus $E = 4 \times 10^4 Pa$, Poisson's ratio $\nu = 0.4$ and density $\rho = 10^3 kg/m^3$. The particles are sampled regularly within each cell by placing each particle on the center of a sub-cell. The FEM box below is with size $4m \times 1m$, Young's modulus $E = 4 \times 10^4 Pa$, Poisson's ratio $\nu = 0.4$ and density $\rho = 10^2 kg/m^2$. The minimal edge length of the FEM mesh and the grid spacing of MPM are with the same value (Δx) in each experiment. The MPM box initially is placed Δx above the FEM box and then simulated under gravity ($g = 10m/s^2$) until no oscillation is observed. To accelerate simulation to reach its final static state, PIC transfer scheme and implicit Euler time integration with large time step sizes (up to CFL limit for MPM) are used. The Newton tolerance is set to $\epsilon_d = 10^{-9} m/s$. The contact stiffness is set to $\kappa = 10^6 Pa$ for all experiments. Figure 4.13 (b) and Figure 4.13 (c) show the final equilibria under low resolution and high resolution respectively.

To examine the convergence rate of displacement to high-resolution results, the example

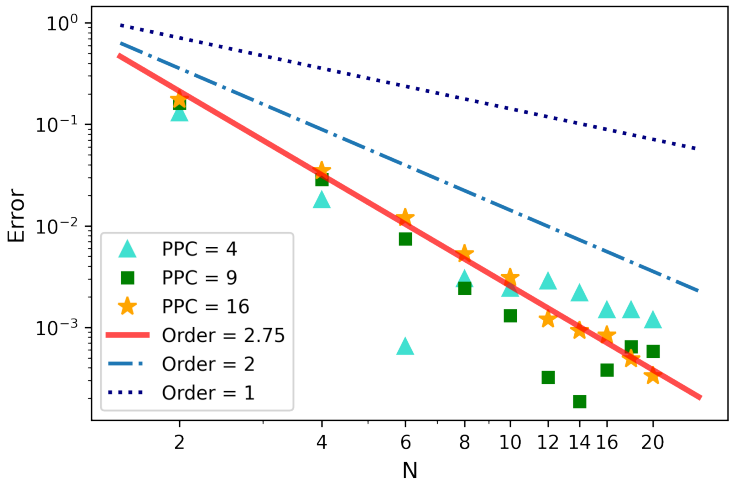


Figure 4.14: **Convergence under Refinement.** Higher PPC can reduce the noise in the convergence curve at higher resolutions. BFEMP with PPC = 16 achieves a convergence order of 2.75 to high-resolution result. Convergence curves with order 1 and order 2 are also plotted for reference.

is refined with $\Delta x = \frac{1}{N}$ and $\hat{d} = \frac{1}{N^2}$, where N iterates all positive even numbers smaller than or equal to 20. The reference high-resolution result is choose as with $N = 30$. The error is defined as the difference in height of the center of mass of the whole domain (with both FEM and MPM domains) between each testing resolution and the high-resolution reference. Due to quadrature error in MPM (de Vaucorbeil et al., 2019), we also experiment with three different particle per cell (PPC) values: 4, 9, and 16. The three error sequences are plotted in Figure 4.14. As observed from the plot, a higher PPC value can reduce the noise in the convergence curve. The error sequence with PPC 16 almost falls into line. Under this setting, BFEMP achieves a convergence order of 2.75.

4.1.6.6 Buckling Behaviours under Different Friction Coefficients

This example tests frictions between two semi-circular rings with large deformation. The two semi-circular rings are stacked together. As the outer semi-circular ring is compressed, different buckling patterns of the inner semi-circular ring under different friction coefficients are observed. This example is modified from the version with FEM-FEM contact in (Zimmerman and Ateshian, 2018).

The experiment setup is shown in Figure 4.15. The outer semi-circular ring with outer radius $14m$ and inner radius $12m$ is discretized by FEM with 2714 vertices and 5067 triangles. The inner semi-circular ring with outer radius $11.99m$ and inner radius $10m$ is discretized by MPM with 10261 particles with grid spacing $0.25m$. The two semi-circular rings are both with Young's modulus $E = 10^6 Pa$, Poisson's ratio $\nu = 0.3$ and density $\rho = 100kg/m^2$. One FEM plate is placed $10^{-3}m$ above the outer semi-circular ring. The displacement of this plate is prescribed to follow a rigid linear motion with a constant downward velocity $1m/s$. Large friction ($\mu = 10$) between the plate and the outer semi-circular ring is activated so that the plate can be viewed as a BFEMP based no-slip boundary condition. The feet of two semi-circular rings are fixed using the level set-based no-slip boundary condition. Another level set-based slip boundary condition is added at the bottom middle below the semi-circular rings to prevent the inner semi-circular rings from colliding into the ground. The contact active distance and the contact stiffness are set to $\hat{d} = 10^{-3}m$ and $\kappa = 10^5 Pa$. The static friction velocity threshold is set to $\epsilon_v = 10^{-5}m/s$. Implicit Euler time integrator with time step size $h = 10^{-2}s$ and Newton tolerance $\epsilon_d = 10^{-4}m/s$ are used.

We vary the friction coefficient between the two semi-circular rings from $\mu = 0$, $\mu = 0.2$ and $\mu = 0.5$. The compression procedure is visualized in Figure 4.15. In the beginning, there is little difference between the three settings. As the FEM plate moves further down, the inner MPM semi ring under the friction-free setting is buckled first as expected. Friction with $\mu = 0.2$ lags the appearance of the buckling. For the large friction case with $\mu = 0.5$, no buckling happens at all.

4.1.6.7 3D Twist with Friction

To test BFEMP's contact and friction model in 3D, a twist test between a FEM spherical shell and an MPM cube is conducted. The FEM shell is controlled to exert a constant twist speed. Under different friction coefficients, it is expected to observe different maximal twist angles on the MPM cube. This example is modified from the version with FEM-FEM contact in (Zimmerman and Ateshian, 2018) as well.

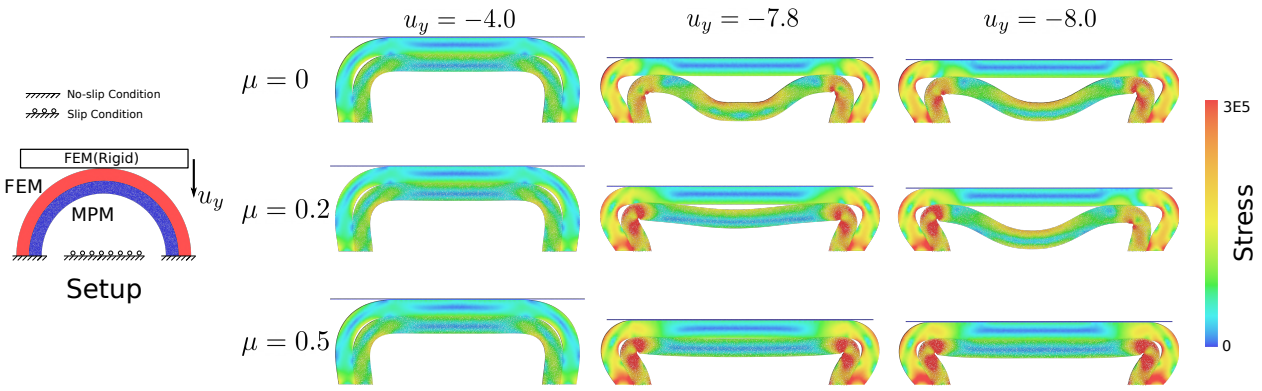


Figure 4.15: **Buckling Behaviours under Different Friction Coefficients.** The experiment setup is illustrated on the left. Under different friction coefficients, the buckling appears at different vertical displacements (u_y).

The initial setup is illustrated in Figure 4.16 (a). The MPM cube with an edge length of $1m$ is placed $10^{-2}m$ below the FEM shell. It is discretized by 90929 particles, where the grid spacing is $0.0625m$. The Young's modulus is $10^8 Pa$. The Poisson's ratio is 0.4. And the density is $100kg/m^3$. The FEM shell with inner radius $0.45m$ and outer radius $0.5m$ is discretized by 1364 points and 3920 tetrahedra. The Young's modulus is $10^{10}Pa$. The Poisson's ratio is 0.4. The density is $10^4 kg/m^3$. The displacements of the top part of the shell are prescribed to follow a rigid motion to exert downward compression and constant-speed twist: it first compresses down with a constant speed $0.5m/s$ for $1s$ (Figure 4.16 (b)) and then rotates around the z-axis with a constant angular velocity $\frac{\pi}{5}$ for $4.5s$. The contact active distance and the contact stiffness are set to $\hat{d} = 10^{-2}m$ and $\kappa = 10^7 Pa$. For settings with frictions, the static friction velocity threshold is set to $\epsilon_v = 10^{-3}m/s$. The simulation uses the implicit Euler time integrator with the time step size $h = 10^{-2}s$. The Newton tolerance is set to $\epsilon_d = 10^{-3}m/s$.

With different friction coefficients, the slipping between the shell bottom and the top center of the cube happens after different twist angles R_z. The final equilibria when the shell stops twisting are visualized in Figure 4.16 (c) (d) (e) (f). The twist angles of the top center part of the cube are plotted in Figure 4.17. Since the pressure forces are between triangles and particles, the interface between the shell and the cube is not perfectly smooth. This roughness results in that the slipping happens when $R_z = 0.1\pi$ in the friction-free

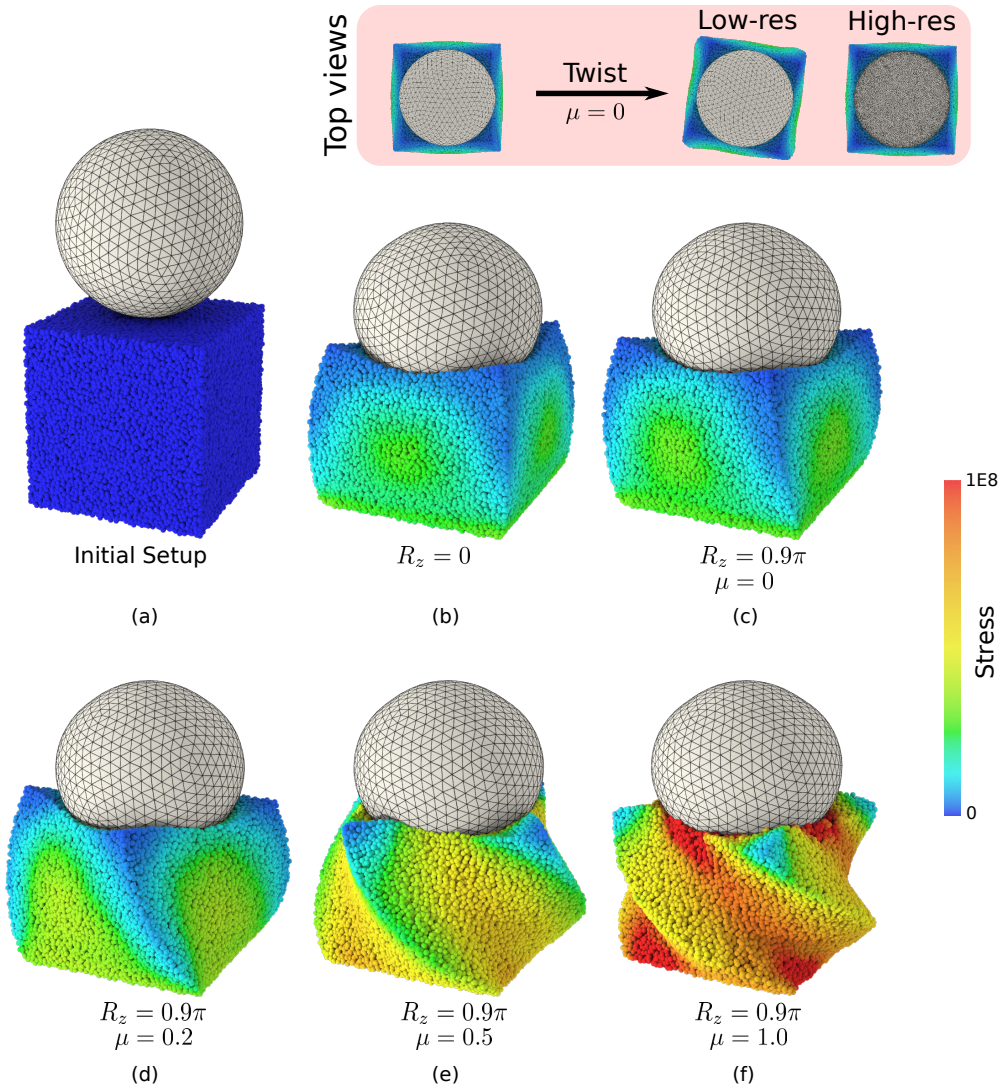


Figure 4.16: **3D Twist with Friction.** (a) Initial setup: The FEM spherical shell is placed $\hat{d}$ above the MPM cube; (b) Before twist procedure, the spherical shell is controlled to press down $0.5m$; (c, d, e, f) Equilibria under different friction coefficients when the shell stops rotating. The nonzero rotation angle with $\mu = 0$ is caused by the non-smoothness of the contacting interfaces, which will decrease as the resolution increases (top views).

settings. The final rotation angle should decrease as the resolution increases. To verify this, we increase the resolution of the FEM mesh and compare the final equilibria in the original setting and the higher-resolution setting. The top views are attached in Figure 4.16, which shows that, with higher resolution, the final state of the cube is close to the initial state before the twisting. For $\mu = 0.2$ and $\mu = 0.5$, the slipping happens when the twists angles are around $R_z = 0.3\pi$ and $R_z = 0.7\pi$ respectively. With $\mu = 1.0$, there is no slipping between

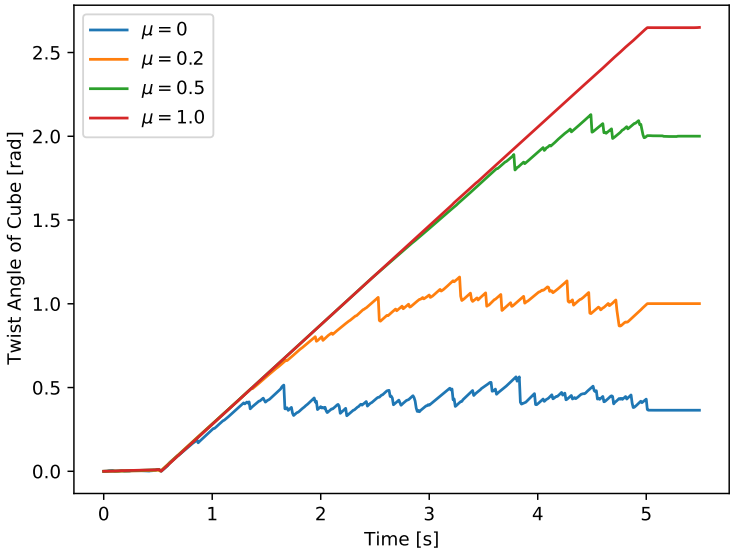


Figure 4.17: **3D Twist with Friction.** The average twist angle around the z-axis of the top center of the cube. The twist procedure happens between $0.5s$ and $5s$. The slipping between the cube and the shell appears at different time points under different friction coefficients.

the shell and the cube.

4.1.7 Conclusion

In this paper, we proposed a new method for monolithically coupling an MPM domain and a FEM domain for elastodynamics through frictional contact. By approximating the non-interpenetration constraint with a barrier energy term and performing time integration using a variational formulation, our method guarantees that no particles will penetrate into the FEM mesh throughout the simulation. Furthermore, when the displacement of the FEM domain is fully prescribed, BFEMP reduces to an explicit mesh-based boundary treatment for MPM. Through numerical experiments validating the energy behavior, robustness, stability, and accuracy, we demonstrated the advantages of the proposed method.

Limitations and Future Works For our current formulation, when MPM particles get very close to the FEM boundary, there is in fact a small portion of overlap between FEM and MPM domains even when there is no interpenetration. This is because MPM particles represent a region of the domain. Although the overlapping area vanishes under spatial

refinement, it would still be interesting to also consider the size and deformation of the regions when defining the distance constraints. In addition, it would be meaningful to extend our framework to support cutting of MPM solids by FEM meshes, where the MPM particles on different sides of the FEM mesh should not communicate with each other even when the FEM mesh is much thinner than the MPM kernel.

4.2 A Dynamic Duo of Finite Elements and Material Points

4.2.1 Introduction

The Finite Element Method (FEM) has achieved notable success in animating elastic objects, such as solids, shells, and rods (Teran et al., 2005; Grinspun et al., 2003; Bergou et al., 2008). Despite its advantages, FEM encounters challenges with severe deformations, often resulting in an ill-conditioned system due to its total Lagrangian nature where the reference configuration is always at the initial time step. Furthermore, handling topology changes, particularly those induced by plasticity, remains a significant hurdle. To overcome these issues, researchers have proposed sophisticated re-meshing techniques (O'Brien et al., 2002; Bargteil et al., 2007). In contrast, the Material Point Method (MPM) employs a particle-based spatial discretization, simplifying the handling of topology changes. The auto-remeshing effect provided by the updated Lagrangian nature of MPM helps maintain a well-conditioned system even under severe deformations, where the reference configuration is at the previous time step. Additionally, MPM's 'plug-and-play' plasticity handling revolutionizes the animation of materials that undergo plastic deformations, such as snow, sand, foam, and fractures (Stomakhin et al., 2013; Ram et al., 2015; Klár et al., 2016; Wolper et al., 2019). However, MPM requires a super-high resolution of particles to represent fine-detailed geometry, making it less efficient for simulating purely elastic objects than FEM, where adaptive meshing can be more effective.

This contrast between FEM and MPM underscores the need for their coupling in complex simulations, combining FEM's precision in geometry and elastic behavior with MPM's

Figure 4.18: **Multi-Material Simulations using Dynamic Duo.** From left to right: a metal boat propelling through water; multicolored sand passing through holes in fabric; honey interacting with a textile surface; and a disaster caused by a debris flow. Each scene highlights the intricate interaction between different materials and structures, emphasizing the fidelity and adaptability of our coupled FEM-MPM simulator.

robustness in handling topological changes and plastic deformations. However, this coupling is not without challenges. FEM typically employs implicit time integration for stability, while explicit integration is favored in MPM, particularly for its ease in implementing plasticity models and because the computational cost of each matrix-vector multiplication in matrix-free implicit MPM is comparable to that of each explicit integration step. The time step sizes in these two integration methods can vary significantly, often by several orders of magnitude. Consequently, asynchronous coupling becomes essential to maintain their respective efficiencies. Another critical challenge is contact handling between the two domains. Contact force modeling is pivotal for two-way coupling, as it is the primary means of communication between the FEM and MPM domains.

To address these challenges, we propose a novel method to couple FEM and MPM. Our approach incorporates an asynchronous time splitting of FEM elasticity, MPM elastoplasticity, and inter-domain frictional contact forces, leveraging the state-of-the-art Incremental Potential Contact (IPC) model (Li et al., 2020) to resolve contact forces between FEM surface triangles and MPM particles. Due to the high stiffness of the contact barrier in IPC, we couple FEM elasticity and inter-domain contact together by implicit integration under a large time step size. Observing the independent interaction of each particle with FEM bodies in this stage, we filter out non-colliding particles and apply a two-stage Newton's method, where elements are frozen once its solution accuracy is achieved, followed by the resolution of per-particle subproblems. After the implicit coupling, MPM elasticity is then explicitly integrated with

a smaller time step size and can be combined with various plasticity models. In this stage, contact forces are treated as constant external forces, and friction integration is stabilized using Coulomb's friction law applied in each substep of MPM based on the current relative tangential velocity. We provide techniques to control penetrations due to time splitting and leverage a closest penetration-free state that is guaranteed to exist for visualization.

In summary, our technical contributions include:

- A novel framework for two-way coupling between meshed finite elements in arbitrary codimensions and meshless material points with arbitrary elastoplastic models.
- An asynchronous time-splitting scheme that effectively integrates implicit FEM and explicit MPM under significantly different time step sizes.
- Numerical treatments to accelerate particle-triangle contact resolutions within the implicit coupling step.
- An IPC-based MPM grid friction model that adheres to Coulomb's friction law.
- Techniques to reduce penetrations from splitting and guarantee penetration-free visualizations.

We demonstrate the effectiveness of our framework by simulating the coupling between FEM soft bodies, rigid bodies, and cloth with a wide range of MPM elastoplastic materials including water, sand, snow, and mud.

4.2.2 Related Work

Finite Element Method Pioneered by Terzopoulos et al. (1987), FEM has established itself as a fundamental technique for modeling elastic bodies in computer graphics. In recent physics-based animation research, robustness and efficiency have been critical. On the local level, robust constitutive models have been explored (Irving et al., 2004, 2006; Smith et al., 2018; Kim et al., 2019). These models accommodate extreme deformations by allowing inverted or degenerated elements. On the global level, advancements have focused

on developing new solvers for the governing nonlinear systems. Teran et al. (2005) introduced a method to project local Hessians to positive definite, thus greatly enhancing the stability of Newton's method. Gast et al. (2015) reformulated the nonlinear system into an optimization problem, enabling the use of line search for guaranteed convergence and allowing frame-rate time step sizes. Bouaziz et al. (2014); Overby et al. (2017) solved the time integration through a local-global alternating minimization while maintaining a fixed global system Hessian. Trusty et al. (2022) employed a mixed variational finite-element formulation and proposed an efficient solver. The domain decomposition technique has also been explored (Li et al., 2019a; Wu et al., 2022). On the other hand, FEM discretization has been successfully applied to co-dimensional objects, such as cloth (Baraff and Witkin, 1998), shells (Grinspun et al., 2003; Chen et al., 2023d), and rods (Bergou et al., 2008), and has been utilized to simulate rigid body dynamics through high-stiffness elasticity (Lan et al., 2022a). Flow-like and brittle materials can also be modeled (O'brien and Hodgins, 1999; Bargteil et al., 2007; Wojtan and Turk, 2008), though frequent remeshing is required to prevent locking artifacts and support topology changes. A fundamental problem in modeling FEM object interactions is contact handling. The state-of-the-art method, Incremental Potential Contact (IPC) (Li et al., 2020), uses a contact barrier to ensure interpenetration-free simulations. This method has been extended to simulate co-dimensional objects (Li et al., 2021a). IPC plays a vital role in our method to resolve FEM self-collisions and FEM-MPM inter-domain collisions.

Material Point Method MPM is a hybrid simulation method that combines Lagragian particles and Eulerian grids. Since its introduction to computer graphics (Hegemann et al., 2013; Stomakhin et al., 2013), it has revolutionized simulations involving large deformations and frequent topology changes. Researchers have focused on designing diverse plasticity models to simulate a variety of dynamic behaviors, including snow (Stomakhin et al., 2013), sand (Klár et al., 2016; Daviet and Bertails-Descoubes, 2016), foam (Yue et al., 2015; Ram et al., 2015), viscoelastic rubber (Fang et al., 2019), phase changes (Stomakhin et al., 2014; Su et al., 2021), and damage (Wolper et al., 2019, 2020; Fan et al., 2022). To overcome the limitations of Eulerian grids to represent detailed geometries, MPM can also be combined

with meshes along with specially designed constitutive models (Fei et al., 2017; Jiang et al., 2017a; Fei et al., 2018; Han et al., 2019; Fei et al., 2019). In parallel, significant efforts have been made to enhance the efficiency of MPM. Fang et al. (2018) explored time-step adaptivity and optimized particle-grid transfers on sparse grids. Gao et al. (2017) introduced adaptive grids. Further optimizations of MPM on GPUs and distributed systems have been achieved by Gao et al. (2018b); Wang et al. (2020b); Qiu et al. (2023); Fei et al. (2021b). While implicit MPM offers guaranteed stability for frame-rate time integration, it requires sophisticated acceleration algorithms, such as multi-grid methods, to tackle the challenges posed by large-scale implicit nonlinear systems with large stencils (Wang et al., 2020a). Additionally, the return mapping generally results in an asymmetric force Jacobian. This asymmetry necessitates intensive, model-by-model mathematical derivations to develop integrable equivalent force formulations (Li et al., 2022g) for robust implicit time integration.

FEM-MPM Coupling The coupling between FEM and MPM has been extensively studied within the mechanical engineering community, driven by a shared motivation with this paper: to combine FEM's efficiency in modeling small deformations due to its use of adaptive meshing and MPM's suitability for simulating large deformations, including fractures. A common approach involved embedding FEM nodes into the MPM grid (Lian et al., 2011b). However, this technique often leads to sticky contact at the FEM-MPM interface, a limitation inherited from MPM. To address this issue, Lian et al. (2011a) developed a separate grid contact model specifically for the interface. Another challenge arises from the requirement for consistent resolutions between the FEM discretization and the MPM grid. The particle-to-surface contact model emerged as an effective solution for this issue (Chen et al., 2015). Despite these advancements, most of these coupling techniques rely on explicit time integration, requiring tiny step sizes for stability and thus overlooking the inherent efficiency of implicit FEM. This limitation has curtailed applications in computer graphics. Alternatively, Li et al. (2022e) explored particle-to-surface IPC to couple implicit FEM with implicit MPM in a monolithic manner. However, this method is confined to elastic objects, diminishing the necessity for coupling, and its computational efficiency is constrained by the implicit MPM bottleneck.

Extending it to support general plasticity encounters similar challenges as those faced by implicit MPM. In contrast, our proposed mixed implicit-explicit time integration not only harnesses the optimal efficiencies of both FEM and MPM but also maintains the flexibility to apply a diverse range of plasticity models.

4.2.3 Governing Equations and Asynchronous Time Splitting

The dynamics of a continuum Ω can be characterized by a time-dependent deformation field $\Phi(\boldsymbol{X}, t)$ from the material space $X \in \Omega$ to its current world space $x \in \Omega^t$ at time t. This map is governed by conservation laws, including mass conservation and momentum conservation:

$$R(\boldsymbol{X}, t)J(\boldsymbol{X}, t) = R(\boldsymbol{X}, 0), \quad R(\boldsymbol{X}, 0)\frac{\partial \boldsymbol{V}}{\partial t}(\boldsymbol{X}, t) = \boldsymbol{f}(\boldsymbol{X}, t), \tag{4.41}$$

where $R(\boldsymbol{X}, t)$ is the mass density field, $J(\boldsymbol{X}, t) = \det \nabla_{\boldsymbol{X}} \Phi(\boldsymbol{X}, t)$ is the Jacobian determinant field, $\boldsymbol{V}(\boldsymbol{X}, t) = \frac{\partial \Phi(\boldsymbol{X}, t)}{\partial t}$ is the velocity field, and $\boldsymbol{f}(\boldsymbol{X}, t)$ is the force density field. Here, we focus on two kinds of internal forces: elastic force, which is defined on the in-domain deformation gradient $\boldsymbol{F} = \nabla_{\boldsymbol{X}} \Phi(\boldsymbol{X}, t)$ and frictional self-contact, which is defined on the domain boundary $\partial\Omega$, and omit external force for simplicity.

To illustrate the asynchronous time splitting techniques for multiple domains, we assume the continuum consists of two disjoint connected components: $\Omega = \Omega_{\mathcal{A}} \cup \Omega_{\mathcal{B}}$. Each domain has its own internal force field $\boldsymbol{f}_{\mathcal{A}}$ and $\boldsymbol{f}_{\mathcal{B}}$ (including elasticity and self-contact). We denote the inter-domain frictional contact force field as $\boldsymbol{f}_{\mathcal{AB}}$. We note that the time splitting is actually used to serialize the action of different forces on the whole domain Ω, so we extend the definition of these forces to the entire domain with zero values.

From t^n to t^{n+1}, we would like to use different time integration schemes (and different spatial discretizations, which will be discussed later) for two elastic force fields: backward Euler for $\boldsymbol{f}_{\mathcal{A}}$ and forward Euler for $\boldsymbol{f}_{\mathcal{B}}$. For stability consideration, the time integration of the frictional contact $\boldsymbol{f}_{\mathcal{AB}}$ is bundled with $\boldsymbol{f}_{\mathcal{A}}$. This leads to the following semi-discrete scheme

for momentum conservation:

$$R^0(\hat{\boldsymbol{V}}^{n+1} - \boldsymbol{V}^n) = h(\hat{\boldsymbol{f}}_{\mathcal{A}}^{n+1} + \hat{\boldsymbol{f}}_{\mathcal{AB}}^{n+1}) \quad \text{(backward Euler)}, \tag{4.42a}$$

$$R^0(\boldsymbol{V}^{n+1} - \hat{\boldsymbol{V}}^{n+1}) = h\boldsymbol{f}_{\mathcal{B}}^{n} \quad \text{(forward Euler)}, \tag{4.42b}$$

where the superscript stands for the discrete time step, $R^0 = R(X, 0)$, $h = t^{n+1} - t^n$ is the time step size and $\hat{\boldsymbol{V}}^{n+1}$ is an intermediate state. Note that Equation (4.42b) has no impact on $\Omega_{\mathcal{A}}$, so we have $\boldsymbol{V}_{\mathcal{A}}^{n+1} = \hat{\boldsymbol{V}}_{\mathcal{A}}^{n+1}$, meaning that there is no extra time integration process on $\Omega_{\mathcal{A}}$. And the equation is simplified to:

$$R_{\mathcal{B}}^0(\boldsymbol{V}_{\mathcal{B}}^{n+1} - \hat{\boldsymbol{V}}_{\mathcal{B}}^{n+1}) = h\boldsymbol{f}_{\mathcal{B}}^{n}. \tag{4.43}$$

On the other hand, there is only contact force $f_{\mathcal{AB}}$ acting on $\Omega_{\mathcal{B}}$ in Equation (4.42a), which leads to

$$R_{\mathcal{B}}^0\hat{\boldsymbol{V}}_{\mathcal{B}}^{n+1} = h\hat{\boldsymbol{f}}_{\mathcal{AB}}^{n+1} + R_{\mathcal{B}}^0 V_{\mathcal{B}}^{n}. \tag{4.44}$$

Then Equation (4.42b) can be rewritten as

$$R_{\mathcal{B}}^0(\boldsymbol{V}_{\mathcal{B}}^{n+1} - \boldsymbol{V}_{\mathcal{B}}^{n}) = h(\hat{\boldsymbol{f}}_{\mathcal{AB}}^{n+1} + \boldsymbol{f}_{\mathcal{B}}^{n}). \tag{4.45}$$

Intuitively, the above equation can be understood that $\hat{\boldsymbol{f}}_{\mathcal{AB}}^{n+1}$ from Equation (4.42a) is treated as a constant external force in Equation (4.42b).

However, forward Euler usually requires much smaller time step sizes compared with backward Euler for stability considerations. This motivates us to use asynchronous time splitting. Assume $\tilde{h} = h/N$ is the time step size required by stability. The one-step forward Euler time integration Equation (4.45) on $\Omega_{\mathcal{B}}$ can be further decomposed into N substeps,

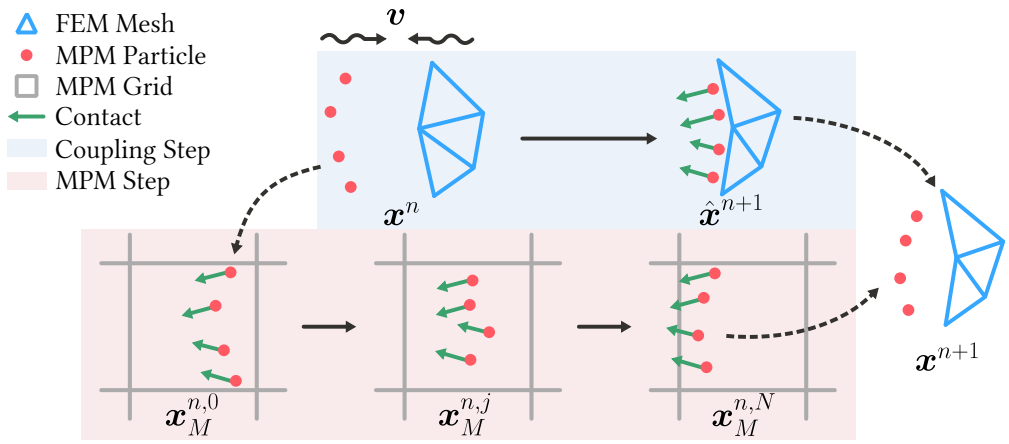


Figure 4.19: **Illustration of the Dynamic Duo.** We couple FEM-MPM inter-domain contact with FEM elasticity by implicit integration. This contact force is then applied as a constant external Lagrangian force on MPM particles throughout N substeps of explicit MPM integration.

leading to the following asynchronous time-splitting scheme:

$$R^0(\hat{\boldsymbol{V}}^{n+1} - \boldsymbol{V}^n) = h(\hat{\boldsymbol{f}}_{\mathcal{A}}^{n+1} + \hat{\boldsymbol{f}}_{\mathcal{AB}}^{n+1}), \tag{4.46a}$$

$$\boldsymbol{V}_{\mathcal{A}}^{n+1} = \hat{\boldsymbol{V}}_{\mathcal{A}}^{n+1}, \tag{4.46b}$$

$$R^0(\boldsymbol{V}_{\mathcal{B}}^{n,j+1} - \boldsymbol{V}_{\mathcal{B}}^{n,j}) = \frac{h}{N}(\hat{\boldsymbol{f}}_{\mathcal{AB}}^{n+1} + \boldsymbol{f}_{\mathcal{B}}^{n,j}), \quad j = 0, 1, 2, ..., N-1, \tag{4.46c}$$

$$\boldsymbol{V}_{\mathcal{B}}^{n+1} = \boldsymbol{V}_{\mathcal{B}}^{n,N}, \quad \boldsymbol{V}_{\mathcal{B}}^{n,0} = \boldsymbol{V}_{\mathcal{B}}^{n}. \tag{4.46d}$$

4.2.4 Dynamic Duo

In this section, we illustrate how two different spatial discretizations, the Finite Element Method (FEM) and the Material Point Method (MPM), work together seamlessly. We show the time-stepping pipeline in Algorithm 4 and in Figure 4.19. Following the convention above, we discretize $\Omega_{\mathcal{A}}$ with FEM meshes and $\Omega_{\mathcal{B}}$ with MPM particles. The inter-domain frictional contact forces are defined between the FEM mesh surface and MPM particles using the Incremental Potential Contact (IPC) method (Li et al., 2020). The significant advantage of explicit MPM, and the primary motivation behind this work, is that explicit MPM can be highly optimized for efficiency and can incorporate different plasticity models without the need for tediously deriving integrable plastic forces (Li et al., 2022g).

Algorithm 4 Dynamic Duo Time Stepping

Scale MPM particle masses by $\frac{2N}{N+1}$; // *Section 4.2.4.4*
Implicit coupling; // *Section 4.2.4.2*
Update FEM states;
Restore MPM particle masses;
// *Explicit MPM step (Section 4.2.4.3):*
Evaluate particle contact forces ($\boldsymbol{f}^{CN}$, $\boldsymbol{f}^{CT}$) and basis velocity $\boldsymbol{v}^B$;
for j = 1, 2, ..., N **do**
    Particle-to-grid transfer of mass, velocity, elasticity, contact, friction, and basis velocity;
    Update grid velocity by explicit integration of elasticity and contact;
    Apply Coulumb's friction law to grid velocity;
    Grid-to-particle transfer to update MPM states;
end for

4.2.4.1 Notations

Let $\boldsymbol{x}_\star, \boldsymbol{v}_\star$ with $\star \in \{F, M\}$ be the nodal positions and velocities of FEM/MPM bodies. Here, $\{\boldsymbol{x}_F, \boldsymbol{v}_F\}$ are defined on FEM mesh vertices, and $\{\boldsymbol{x}_M, \boldsymbol{v}_M\}$ are defined on MPM particles. $\boldsymbol{x} = [\boldsymbol{x}_F, \boldsymbol{x}_M], \boldsymbol{v} = [\boldsymbol{v}_F, \boldsymbol{v}_M]$ are their concatenations. A superscript n can be appended to distinguish different time steps. Viewing the initial positions $\boldsymbol{X} = \boldsymbol{x}^0$ as the material space, $\boldsymbol{x}^n$ is the approximation of $\Phi(\boldsymbol{X}, t^n)$, and $\boldsymbol{v}^n$ is the approximation of $\frac{\partial}{\partial t}\Phi(\boldsymbol{X}, t^n)$. Let $\boldsymbol{M} = \mathrm{Diag}(\boldsymbol{M}_F, \boldsymbol{M}_M)$ be the global diagonal lumped mass matrix formed by integrating $R(X, t)$ over individual FEM elements or MPM particles, $\{\hat{\boldsymbol{x}}, \hat{\boldsymbol{v}}\}$ be the intermediate penetration-free state from the coupling step guaranteed by IPC, and $h_M, h = Nh_M$ be the time step sizes for MPM and FEM, respectively. We distinguish $\hat{\boldsymbol{x}}$ and $\boldsymbol{x}$ because, after the MPM integration, $\boldsymbol{x}_M$ may penetrate into $\boldsymbol{x}_F$, as discussed in Section 4.2.4.4. In addition, $\hat{\boldsymbol{x}}$ serves as a feasible initial guess for the coupling step and the state at which to evaluate friction basis in the IPC model.

4.2.4.2 Implicit Coupling Step

In this step, we conceptualize MPM particles as discrete rigid spheres with radius r excluding self-contact. The contact acts in a thin layer enveloping the sphere (Li et al., 2021a). We take $r = \Delta x/\sqrt[3]{\mathrm{PPC}}$ where Δx is the spacing of the MPM background grid, and PPC stands for particle number per cell. This allows overlaps between particles to prevent unrealistic

penetrations of sharp FEM parts into MPM bodies. After spatial discretization, the elastic and contact forces in the momentum equation (Equation (4.46a)) are defined w.r.t. the positions of vertices and particles, necessitating a time discretization of $\frac{\partial}{\partial t}\boldsymbol{\Phi}(\boldsymbol{X},t)=\boldsymbol{V}(\boldsymbol{X},t)$. Employing the backward-Euler method, the spatially integrated governing equations are discretized as follows:

$$\begin{aligned}&\boldsymbol{M}(\hat{\boldsymbol{v}}^{n+1}-\boldsymbol{v}^{n})=h(\boldsymbol{f}^{E}(\hat{\boldsymbol{x}}_{F}^{n+1})+\boldsymbol{f}^{SC}(\hat{\boldsymbol{x}}_{F}^{n+1})+\boldsymbol{f}^{C}(\hat{\boldsymbol{x}}^{n+1})+\boldsymbol{M}\boldsymbol{g}),\\&\hat{\boldsymbol{x}}^{n+1}=\boldsymbol{x}^{n}+h\hat{\boldsymbol{v}}^{n+1}.\end{aligned}\tag{4.47}$$

Here, the elasticity force $\boldsymbol{f}^{E}$ is represented by the negative gradient of elastic strain energies. This includes various forms of energy: volumetric elasticity energy on tetrahedra for modeling soft bodies; membrane and bending energies on triangles for thin shells; and rigidity energy defined per body to model nearly rigid objects (Lan et al., 2022a). The self-contact $\boldsymbol{f}^{SC}$ and the inter-domain contact $\boldsymbol{f}^{C}$, defined among MPM particles and FEM surfaces, are derived from the negative gradients of frictional contact potentials (Li et al., 2020).

Following (Li et al., 2020), the governing nonlinear equation system can be integrated into an optimization problem w.r.t. nodal positions:

$$\hat{\boldsymbol{x}}^{n+1}=\operatorname{argmin}_{\boldsymbol{x}}\frac{1}{2}\|\boldsymbol{x}-\tilde{\boldsymbol{x}}^{n+1}\|_{\boldsymbol{M}}^{2}+h^{2}(\Psi(\boldsymbol{x}_{F})+B(\boldsymbol{x})-\boldsymbol{x}^{\top}\boldsymbol{M}\boldsymbol{g}).\tag{4.48}$$

Here, $\tilde{\boldsymbol{x}}^{n+1}=\boldsymbol{x}^{n}+\boldsymbol{v}^{n}h$ represents the predictive position under inertia, Ψ denotes the elastic potential of FEM bodies, and B is the frictional contact potential. Notably, after the MPM step in the previous time step, there might be slight penetrations in $\boldsymbol{x}^{n}$. So we use $\hat{\boldsymbol{x}}^{n}$ instead of $\boldsymbol{x}^{n}$ as the starting configuration for the optimization. After solving this optimization, the positions and velocities of FEM vertices are updated accordingly to $\boldsymbol{x}_{F}^{n+1}=\hat{\boldsymbol{x}}_{F}^{n+1}$, $\boldsymbol{v}_{F}^{n+1}=\hat{\boldsymbol{v}}_{F}^{n+1}$. It is important to note that MPM particles are not advected in this step to avoid inconsistencies between tracked deformation gradients and particle positions. This optimization problem is solved using Newton's method with a backtracking line search, where the initial step size is determined by continuous collision detection (CCD) to prevent penetration during the optimization process (Li et al., 2020). When assembling the global

Hessian matrix, local Hessians are projected to be semi-positive definite to guarantee an energy-decreasing direction. The linear system is solved by the Conjugate Gradient (CG) method preconditioned by the 3×3 diagonal blocks.

Non-colliding Particle Filtering Due to the inherent nature of MPM, the number of MPM particles significantly exceeds the number of FEM vertices. However, a large proportion of particles do not even collide with FEM bodies during a time step. For these non-colliding particles, their next state of non-penetration, $\hat{\boldsymbol{x}}^{n+1}$, can be analytically determined as $\tilde{\boldsymbol{x}}^{n+1}$, and they do not interfere with other particles or FEM bodies. To optimize computational resources, we can safely exclude these non-interacting degrees of freedom in Equation (4.48). To do the filtering, we only keep particles that have potential collisions as the scene moves from $\boldsymbol{x}^n$ to $\boldsymbol{x}^n + 2h(\boldsymbol{v}^n + \boldsymbol{g}h)$. The collisions are detected by checking overlaps between trajectories' bounding boxes.

Two-stage Newton's Method To further accelerate convergence, we employ a two-stage Newton's method. In the first stage, we solve the full nonlinear optimization until the residual on FEM DOFs reaches the desired tolerance. We then freeze FEM bodies and continue to optimize the particle DOFs. A key observation is that once the FEM domain is fixed, the entire optimization problem can be effectively divided into independent sub-problems for each particle. However, direct per-particle optimization is not trivial to implement on a GPU since the contact pairs may vary over time. Instead, we still simultaneously search for all particles, but with several acceleration techniques:

- The system matrix is now 3×3 block-diagonal, consisting only of the diagonal mass matrix and the diagonal blocks of the barrier Hessian. The inverse of the Hessian can then be efficiently evaluated per diagonal block.
- For the backtracking line search, we perform CCD to clamp the search directions per particle and then only halve per-particle step sizes on energy-increasing particles.
- We continue to freeze particles that reach the desired accuracy because of the indepen-

dence of particles.

4.2.4.3 Explicit MPM Step

Due to the asynchronous time splitting, each time step comprises N sub-steps of explicit MPM integration. At the end of the coupling step, the IPC force $\boldsymbol{f}^C$ is evaluated on particles, which is decomposed as the sum of a normal contact force $\boldsymbol{f}^{CN}$ and a tangential friction force $\boldsymbol{f}^{CT}$. $\boldsymbol{f}^{CN}$ is then treated as a constant Lagrangian external force applied to the particles. $\boldsymbol{f}^{CT}$, requiring special consideration, will be discussed in a separate section. We follow MLS-MPM (Hu et al., 2018a) for our explicit MPM sub-stepping. Each particle's state is described by a four-tuple $(\boldsymbol{x}_p, \boldsymbol{v}_p, \boldsymbol{C}_p, \boldsymbol{F}_p^E)$: $\boldsymbol{x}_p$ denotes the particle position, $\boldsymbol{v}_p$ the particle velocity, $\boldsymbol{F}_p^E$ the elastic deformation gradient tracked on the particle, and $\boldsymbol{C}_p$ the angular momentum matrix (Jiang et al., 2015a). Time integration within MPM occurs on a background grid. At each substep j, from t^n to t^{n+1}, particle mass and velocity are transferred to the grid:

$$m_i^j = \sum_p m_p w_{ip}^j, \quad \boldsymbol{v}_i^j = \frac{1}{m_i^j} \sum_p w_{ip}^j m_p (\boldsymbol{v}_p^j + \boldsymbol{C}_p^j (\boldsymbol{x}_i - \boldsymbol{x}_p^j)), \tag{4.49}$$

where x_i is the position of grid node i, w_{ip}^j represents a quadratic MLS basis defined at grid node i evaluated at $\boldsymbol{x}_p^j$, m_i^j is the transferred grid node mass, and v_i^j is the transferred grid velocity. For simplicity, we have omitted the superscript n in Equation (4.46c). The discretized momentum equation on the grid, Equation (4.46c), is expressed as $\boldsymbol{v}_i^{j+1} = \boldsymbol{v}_i^j + h_M \boldsymbol{f}_i^j / m_i^j$, where the grid $\boldsymbol{f}_i^j$ is the sum of the transferred normal contact force $\boldsymbol{f}_i^{CN,j} = \sum_p w_{ip}^j \boldsymbol{f}_p^{CN}$, gravity force $m_i^j \boldsymbol{g}$ and elasticity force $\boldsymbol{f}_i^{E,j} = \sum_p V_p^0 \boldsymbol{\tau}(\boldsymbol{F}_p^{E,j}) \nabla w_{ip}^j$, where $\boldsymbol{\tau}$ is the Kirchhoff stress. In MLS-MPM, the gradient of the weight function is calculated as $\nabla w_{ip}^j = \frac{4}{\Delta x^2} w_{ip}^j (\boldsymbol{x}_i - \boldsymbol{x}_p^j)$. The updated grid velocities are then transferred back to the particles, updating their states:

$$\begin{aligned} \boldsymbol{v}_p^{j+1} &= \sum_i \boldsymbol{v}_i^{j+1} w_{ip}^j, \quad \boldsymbol{x}_p^{j+1} = \boldsymbol{x}_p^j + h_M \boldsymbol{v}_p^{j+1}, \\ \boldsymbol{C}_p^{j+1} &= \sum_i \boldsymbol{v}_i^{j+1} \nabla w_{ip}^{j,\top}, \quad \boldsymbol{F}_p^{E,j+1} = (\boldsymbol{I} + h_M \boldsymbol{C}_p^{j+1}) \boldsymbol{F}_p^{E,j}. \end{aligned} \tag{4.50}$$

Incorporating plasticity, at the end of each MPM substep, we further pull $\boldsymbol{F}_p^{E,j+1}$ back into a predefined elastic region using the associated return mapping $\boldsymbol{F}_p^{E,j+1} \leftarrow \mathcal{Z}(\boldsymbol{F}_p^{E,j+1})$ (Jiang et al., 2016).

MPM Friction In accordance with physical principles, the friction force should always oppose the relative tangential velocity without altering its direction. Naively applying the evaluated tangential friction force $\boldsymbol{f}^{CT}$ on particles can easily violate this law, causing high-frequency vibration of MPM objects that should remain stationary. To stabilize friction integration, we transfer the basis velocity (defined as the nearby FEM surface velocity) onto the grid. This basis velocity is estimated at the coupling solve's convergence by interpolating the velocities of the friction basis onto MPM particles and then transferred onto grid to serve as the basis velocity for grid nodes:

$$\boldsymbol{v}_p^B = \frac{\sum_{k\in T} \lambda_{k,p} \boldsymbol{v}_k}{\sum_{k\in T} \lambda_{k,p}}, \quad \boldsymbol{v}_i^{B,j} = \frac{\sum_p w_{ip}^j \boldsymbol{v}_p^B}{\sum_{p,\boldsymbol{f}_p^{CT}\neq 0} w_{ip}^j}. \tag{4.51}$$

Here T is the set of contact pairs contributing to friction, $\lambda_{k,p}$ denotes the magnitude of the normal contact force, and $\boldsymbol{v}_k$ is the velocity at the closest point to particle p on the contacting triangle. The need for interpolation arises from the presence of multiple contact pairs that collectively contribute to the total friction force $\boldsymbol{f}^{CT}$ on particle p. Note that this averaging process only includes particles experiencing nonzero friction force, and we skip node i if the denominator is zero. We define the tangential relative velocity on the grid as:

$$\boldsymbol{v}_i^{\text{rel},j+1} = (\boldsymbol{I} - nn^\top)(\boldsymbol{v}_i^{j+1} - \boldsymbol{v}_i^{B,j}), \tag{4.52}$$

where n is the normalized $\boldsymbol{f}_i^{CN,j}$ and $\boldsymbol{v}_i^{j+1}$ is the velocity after applying elasticity and normal contact. The final grid velocity, as adjusted by Coulomb's friction model, is given by:

$$v_i^{j+1} \leftarrow v_i^{j+1} - \min\{\|\Delta \boldsymbol{v}_i^{CT,j}\|, \|\boldsymbol{v}_i^{\text{rel},j+1}\|\}\boldsymbol{v}_i^{\text{rel},j+1}/\|\boldsymbol{v}_i^{\text{rel},j+1}\|, \tag{4.53}$$

where $\Delta \boldsymbol{v}_i^{CT,j} = \boldsymbol{f}_i^{CT,j} h_M / m_i$ is the velocity increment resulting from $\boldsymbol{f}_i^{CN}$ if treated as an external force. Note that the friction coefficients are already utilized to evaluate $\boldsymbol{f}^{CT}$, which may be assembled from interfaces with different friction coefficients. The projection can be understood that the application of $\boldsymbol{f}_i^{CT,j}$, clamped by $\|\boldsymbol{v}_i^{\text{rel},j+1}\|$. This approach effectively ensures that the friction force opposes the relative velocity direction and does not change it. Our handling of friction represents a balance between conserving momentum and maintaining stability, with the latter being more crucial for visual effects. Note that the above friction-related quantities on particles are evaluated at $\hat{\boldsymbol{x}}$.

4.2.4.4 Reducing Penetrations from Splitting

Using a first-order scheme, such as symplectic Euler and backward Euler, the integration of $\frac{D}{Dt}\boldsymbol{V}(\boldsymbol{X}, t) = \boldsymbol{a}$ with a constant acceleration $\boldsymbol{a}$ will yield the same velocity despite varying time step sizes. However, this consistency does not extend to the integration of $\frac{D}{Dt}\boldsymbol{\Phi}(\boldsymbol{X}, t) = \boldsymbol{V}(\boldsymbol{X}, t)$. It is a common observation that, under constant gravity acceleration and using a first-order scheme, objects fall faster with larger time step sizes. This mismatch contributes to the penetrations of MPM particles into FEM bodies. Higher-order schemes may be employed to reduce this mismatch, but they complicate the implementation. Instead, we stick to backward Euler and symplectic Euler coupling for implementation simplicity but propose methods to reduce penetrations due to splitting.

Let the evaluated contact acceleration on a particle be $\boldsymbol{a}$ at coupling step convergence. Ignoring elasticity, from t^n to t^{n+1}, the trial velocity and the final velocity are the same: $\hat{\boldsymbol{v}}^{n+1} = \boldsymbol{v}^{n+1} = \boldsymbol{v}^n + h\boldsymbol{a}$. However, this does not apply to positions. After implicit time integration, the penetration-free position is $\hat{\boldsymbol{x}} = \boldsymbol{x}^n + h\boldsymbol{v}^n + h^2\boldsymbol{a}$. In contrast, using symplectic Euler (ignoring elasticity) with a time step of $\frac{h}{N}$, the final position is $\boldsymbol{x}^{n+1} = \boldsymbol{x}^n + \frac{h}{N}\sum_{j=1}^{N}(\boldsymbol{v}^n + \frac{jh}{N}\boldsymbol{a}) = \boldsymbol{x}^n + h\boldsymbol{v}^n + \frac{N+1}{2N}h^2\boldsymbol{a}$.

One effective way to reduce the mismatch is to decrease h, as the error is $O(h^2)$. The error order might be slightly lower in practice due to the splitting of MPM elasticity and smoothing from particle-grid transfers. Additionally, we employ a plug-and-play mass scaling

mechanism and visualize the closest non-penetration states to eliminate penetration artifacts.

Mass Scaling The principle behind this mechanism is to slightly increase the contact force, repelling MPM solids further away from FEM solids. To reduce penetrations in $\boldsymbol{x}^{n+1}$, we can scale the contact force f by $\frac{2N}{N+1}$ in the MPM step, to align new $\boldsymbol{x}^{n+1}$ with before-scaling $\hat{\boldsymbol{x}}$. However, scaling f during the implicit coupling is also necessary to ensure FEM receives an equivalent momentum correction. This is challenging since f is an implicit force in the coupling step. A solution comes from our discovery that a particle's contact force at equilibrium is approximately proportional to its mass. We scale the particle masses by $\frac{2N}{N+1}$ when solving the implicit coupling, and then restore the original masses during MPM integration. Note that the contact force is still an internal force, not interfering with the total momentum. With this scaling, there will be a greater gap between two domains in $\hat{\boldsymbol{x}}$, leading to fewer penetrations in $\boldsymbol{x}^{n+1}$.

Non-penetration External State To visually address penetrations, we use a separate external state of particles $\boldsymbol{x}_M^{o,n+1}$ solely for rendering purposes. $\hat{\boldsymbol{x}}_M^{n+1}$ lags one time step behind the current state, while the current state $\boldsymbol{x}_M^{n+1}$ may have penetrations. So we visualize $\boldsymbol{x}_M^{o,n+1}$ by freezing FEM solids and performing a per-particle CCD from $\hat{\boldsymbol{x}}_M^{n+1}$ to $\boldsymbol{x}_M^{n+1}$ to find a closest non-penetration state. We only compute an external state per frame for visualization because this state is not used in time stepping.

4.2.5 Evaluation

We implemented our framework on a workstation with an NVIDIA RTX 6000 Ada GPU and an Intel Core i9-10920X CPU. The relative error of CG is set to 10^{-3}. We stop Newton's method if the L_∞ norm of the search direction reaches below $10^{-2}h$ m.

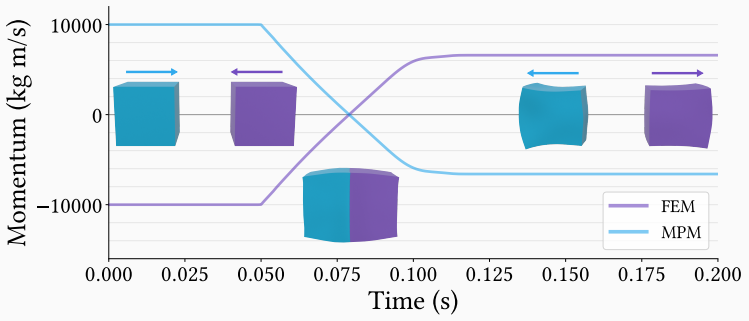


Figure 4.20: Momentum conservation test.

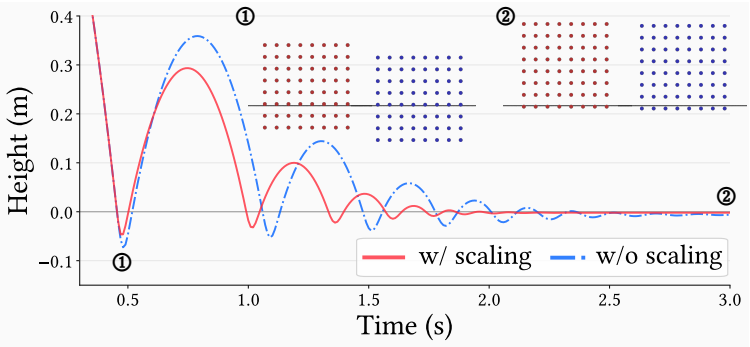


Figure 4.21: Ablation on our mass scaling mechanism.

4.2.5.1 Ablation Studies and Unit Tests

Momentum Conservation. The system's linear momentum is conserved in the absence of friction and external forces. To demonstrate this, we conduct an experiment involving two cubes with identical properties but different discretizations. These cubes collide under the same velocity magnitudes. As shown in Figure 4.20, the total linear momentum remains consistently zero. However, we note that the linear momentum under friction will not be conserved due to our friction clamping mechanism, and the angular momentum is not conserved due to the choice of backward Euler.

Reducing Penetrations. Our mass scaling mechanism can effectively reduce the penetration of the MPM internal state into FEM bodies. In a 2D experiment, an elastic cube is allowed to free fall from a height of 1 meter onto the ground with a relatively large time step size, $h = 10^{-2}$s. As depicted in Figure 4.21, we compare scenarios with and without mass scaling. The results reveal the mechanism's efficacy in reducing penetration. Additionally, a

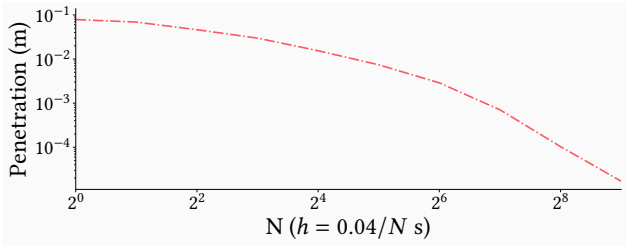


Figure 4.22: Convergence under time step size refinement.

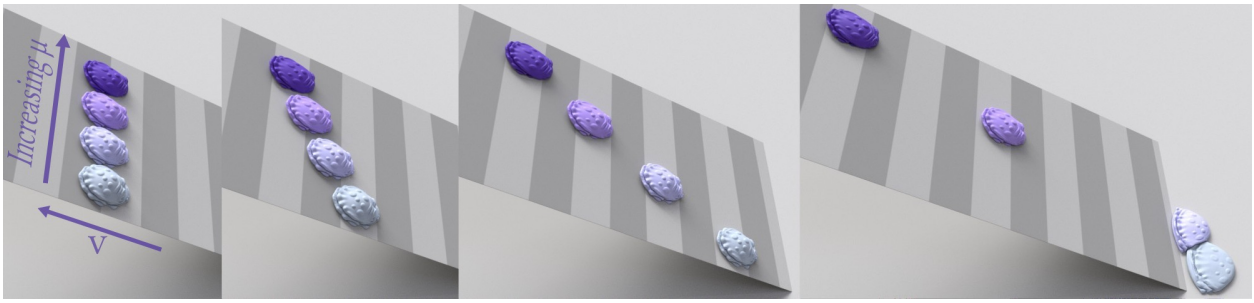


Figure 4.23: **MPM Friction.** MPM Play-Dohs with different friction coefficients on an incline conveyor.

notable side effect of mass scaling is its contribution to stabilizing objects on the ground.

Furthermore, reducing the time step size is another effective way to mitigate penetrations. Under the same experimental setup above, we progressively refined the time step size by a factor of 2. Figure 4.22 demonstrates that penetration can effectively converge to zero with this continuous refinement of the time step size.

MPM Friction. We conduct a 2D experiment to validate our friction projection on the MPM grid. A rectangular elastic object is placed on a horizontal plane under downward gravity. Despite utilizing a small time step size ($h = 10^{-3}$s), the object vibrates on the plane

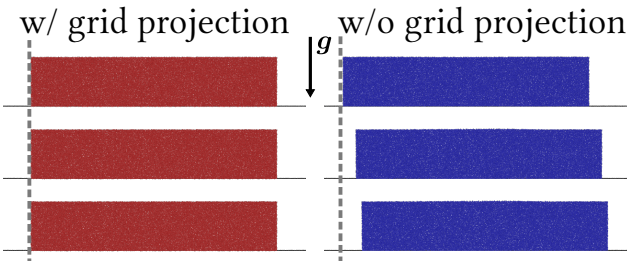


Figure 4.24: Ablation on the MPM friction projection.

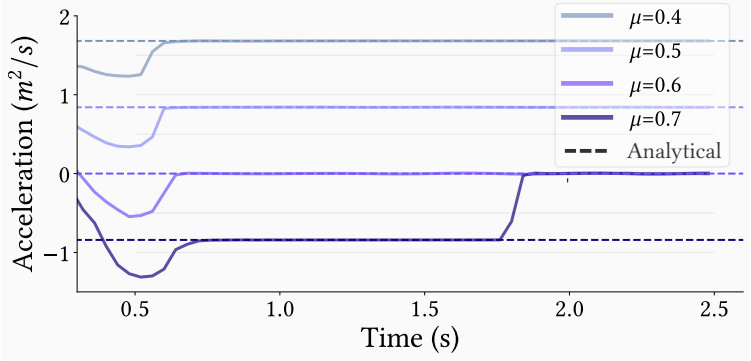


Figure 4.25: Our MPM friction model can resolve static and dynamic friction accurately.

if we directly apply friction as a constant external force, as shown on the right of Figure 4.24. However, by applying our projection on the MPM grid, we effectively address this issue, shown on the left of Figure 4.24.

To further assess the accuracy of our friction model, we conduct an experiment where a conveyor with an inclined angle $\theta = \arctan(0.6)$ tries to move 4 MPM Play-Dohs upward, as shown in Figure 4.23. Each body, with von Mises plasticity, is assigned a distinct friction coefficient ($\mu = 0.4, 0.5, 0.6, 0.7$). The experiment confirms that our model can resolve both static and dynamic friction even on moving interfaces. Figure 4.25 reveals that the accelerations from the dynamic friction align with the analytical solution $\boldsymbol{g}(\sin\theta^* - \mu\cos\theta^*)$ (downward positive direction, $\theta^* = \arctan\mu$). We successfully capture the transition from dynamic to static friction when $\mu = 0.7$ as the relative velocity vanishes.

FEM Boundary in MPM In traditional MPM, boundary conditions are typically enforced via fuzzy, grid-based collision detection, which may overlook fine geometrical details close to the grid resolution. In contrast, CCD in our method can resolve collisions with objects of any thickness, independent of the MPM grid resolution. Illustrated in Figure 4.26, we compare our approach with traditional MPM in a scenario where a mass of viscous fluid falls onto a thin wire mesh. With our method, the fluid successfully adheres to the wires due to friction, whereas traditional MPM fails to capture the interaction between the fluid and the wire mesh.

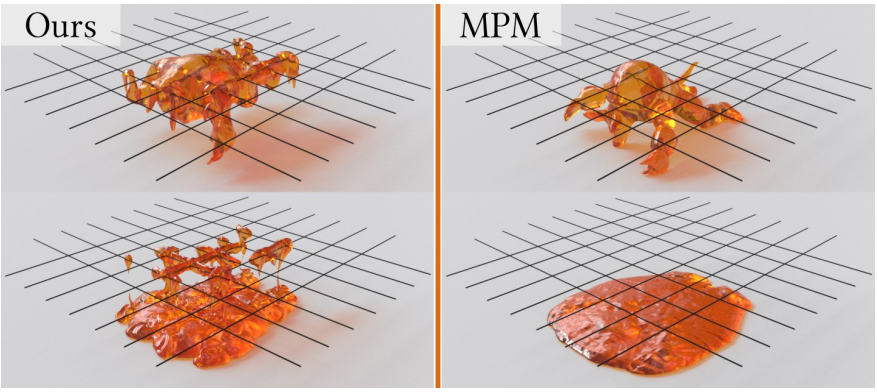


Figure 4.26: Our method can resolve thin collision objects represented by FEM meshes in MPM simulations.

Table 4.4: Simulation statistics of our multi-material simulation demos.

	h (s)	h_M (s)	s/step	#P ($\times 10^6$)	#V ($\times 10^3$)
Boat	2e-3	5e-5	3.81	3.12	10.2
Ruts	2e-3	2e-5	3.70	2.80	38.0
Dough	2e-5	2e-5	0.67	0.727	10.7
Snowball	5e-3	5e-5	2.22	2.14	2.66
Honey	2e-3	1e-5	1.72	1.19	29.2
Colored Sand	2e-3	2e-5	11.7	5.57	38.4
Debris Flow	5e-3	5e-5	6.71	1.47	236

4.2.5.2 Multi-Material Simulation

In this section, we conduct a comprehensive evaluation of the Dynamic Duo on its performance in two-way coupled simulations involving a diverse set of FEM elastic solids, such as soft bodies, rigid bodies, cloth, and various MPM elastoplastic materials, including fluid, sand, snow, and debris flow. Detailed simulation statistics can be found in Table 4.4. The average timing per step is reported. The last two columns are the particle count and the vertex count.

Boat In Figure 4.27, we demonstrate the use of affine body dynamics (Lan et al., 2022a) to simulate the movement of a rigid boat through a tank filled with J-based MPM fluid (Jiang et al., 2015a). The propeller is attached to the shaft through contact. Buoyancy keeps the boat afloat. The interaction between the propeller and the water generates forward thrust by pushing the water backward. The propeller is not controlled by Dirichlet boundary conditions.

Figure 4.27: **Boat.** An FEM boat's progression through MPM water.

Instead, the effect of motor power on the propeller is mimicked by a balanced rotational external force field to generate a naturalistic interaction between the propeller and the boat.

Ruts In Figure 4.28, we present an experiment, inspired by Zhao et al. (2023a), to demonstrate our Dynamic Duo's potential applications in geotechnical engineering, particularly in analyzing soil interactions. We simulate a scenario of a NASA Mars rover's wheel traversing granular soil modeled using MPM with Drucker-Prager plasticity, thus leaving deep ruts. Similar to the previous example, the wheel's movement is driven by a balanced rotational external force field. Notably, we haven't introduced additional friction between the soil and the wheel; instead, the thrust is generated solely through the friction amongst the soil grains themselves.

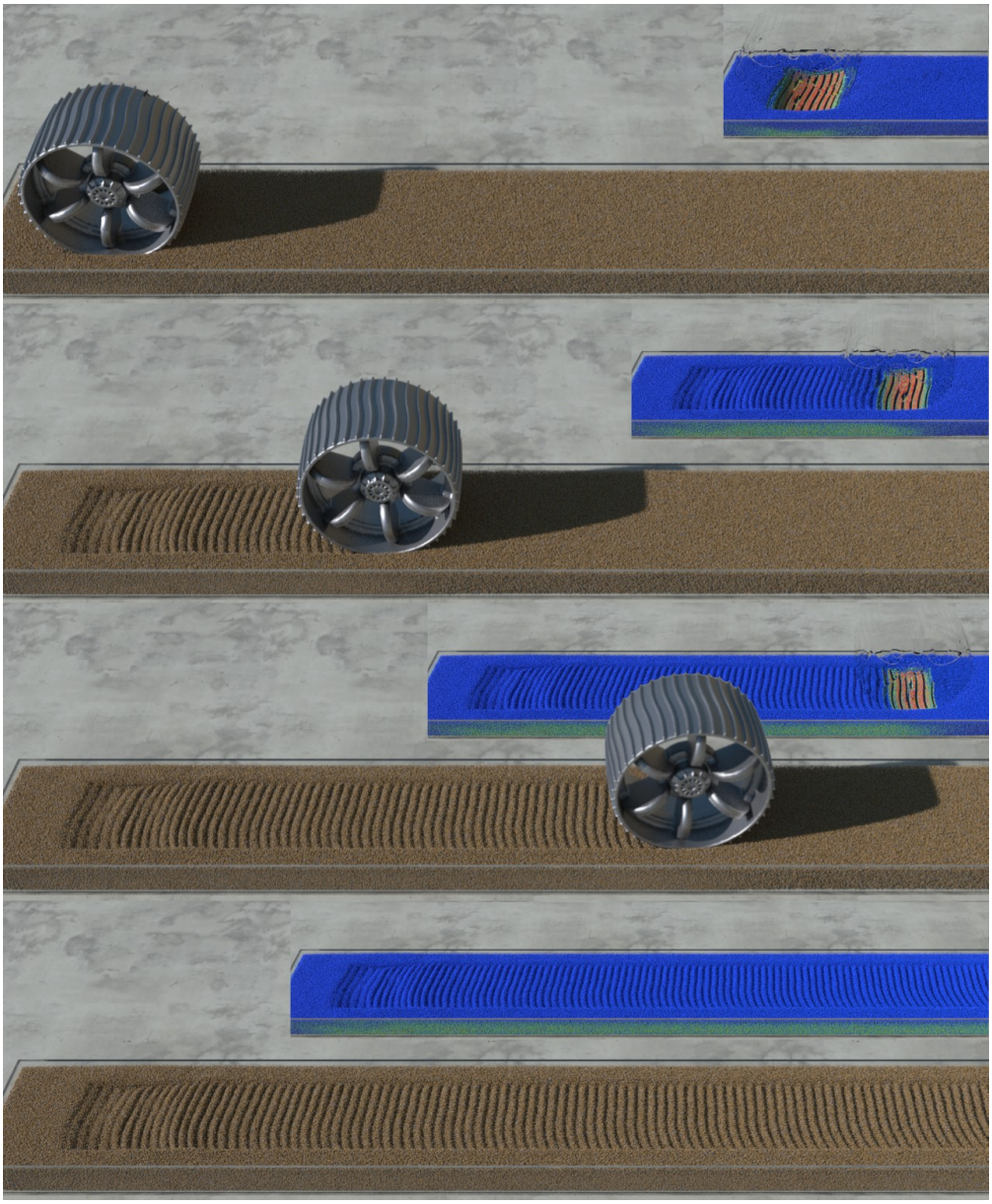


Figure 4.28: **Ruts.** The wheel of a Mars rover navigates through the soil and leaves deep ruts in its path. We also visualize the contact force on soil particles, indicated by a color gradient ranging from blue (low) to red (high).

Dough In Figure 4.29, we simulate a common kitchen task to demonstrate the application of rigid bodies in soft body manipulation, inspired by (Huang et al., 2021). A rolling pin flattens MPM dough with von Mises plasticity. The rolling pin's handle follows a predefined path, while the roller is attached around it by contact, free to rotate. The two-way interaction is indicated by the rotation driven by friction between the dough and the roller.

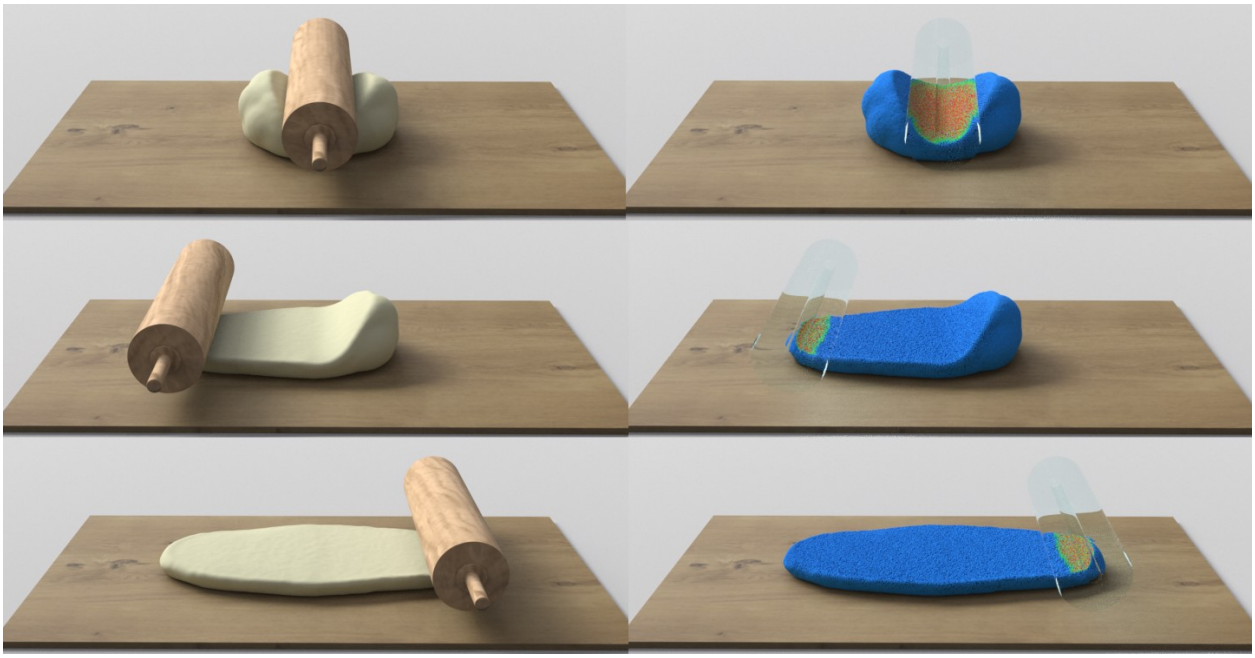


Figure 4.29: **Dough.** A rolling pin rolls out an MPM dough. Contact force is visualized on particles by a color gradient ranging from blue (low) to red (high).

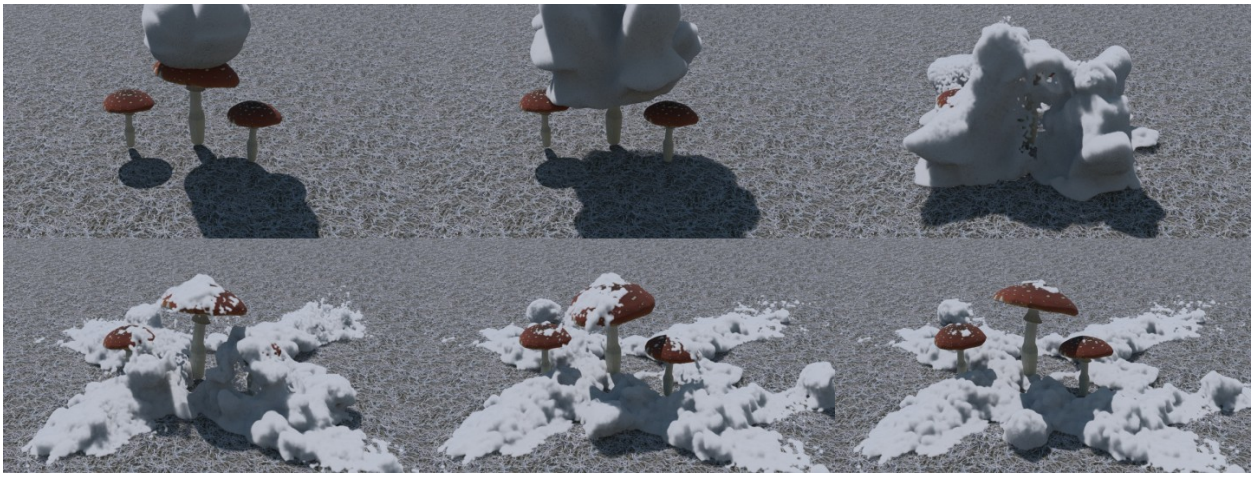


Figure 4.30: **Snow.** Soft FEM mushrooms undergoing elastic deformations by the impact of a falling MPM snowball.

Snowball In Figure 4.30, we drop an MPM snow ball modeled with Cam-Clay plasticity (Gaume et al., 2018) onto soft FEM mushrooms. The two-way impact smashes the snowball and also deforms the elastic mushrooms. Additionally, due to friction, portions of the snow adhere to the tops of the mushrooms. Notably, as detailed in Figure 4.31, the computational cost of contact handling (red) in this scenario is relatively moderate compared to MPM time

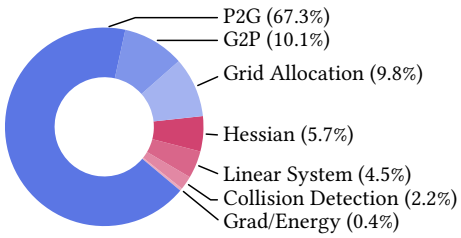


Figure 4.31: The timing pie chart of the snowball example.

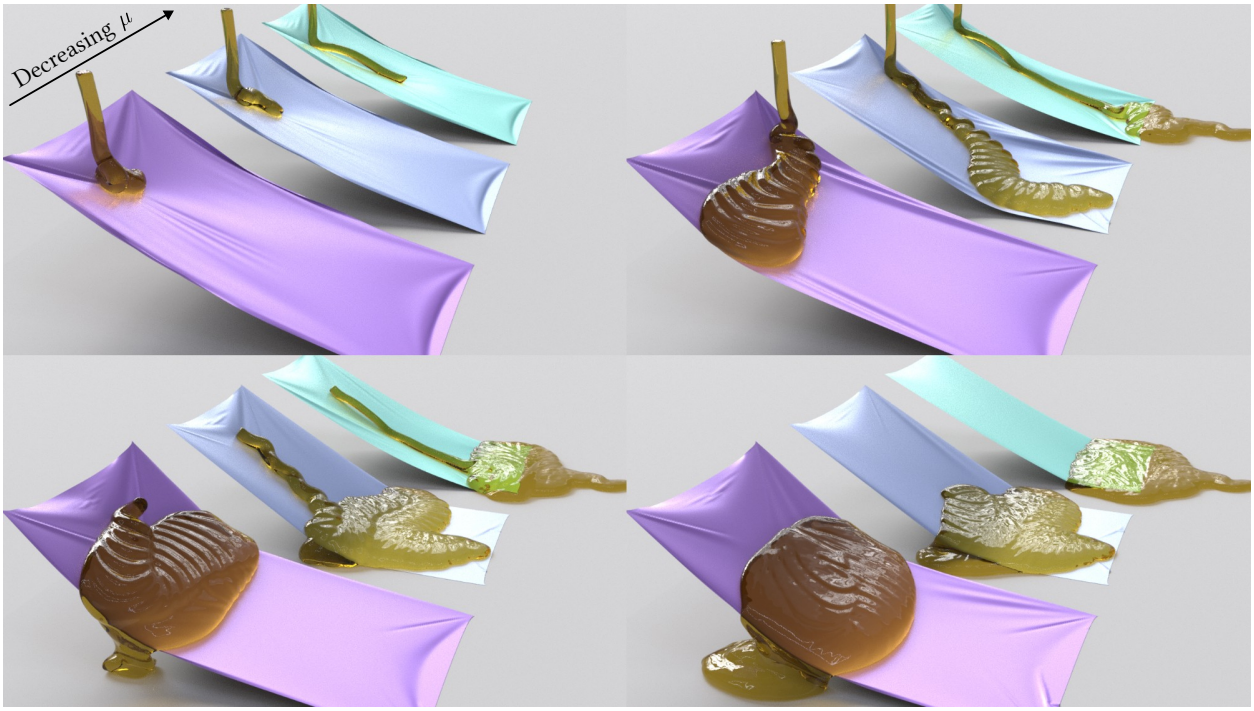


Figure 4.32: **Honey.** Different buckling patterns of honey on a piece of cloth with different friction coefficients.

integration (blue).

Honey In Figure 4.32, we present a simulation where honey is poured over pieces of cloth, each having different friction coefficients. The dynamics of the cloth are captured using ARAP membrane energy and dihedral bending energy (Grinspun et al., 2003), and the honey is modeled as MPM J-based fluids with viscosity. The varying magnitudes of friction result in distinct buckling patterns on the cloth.

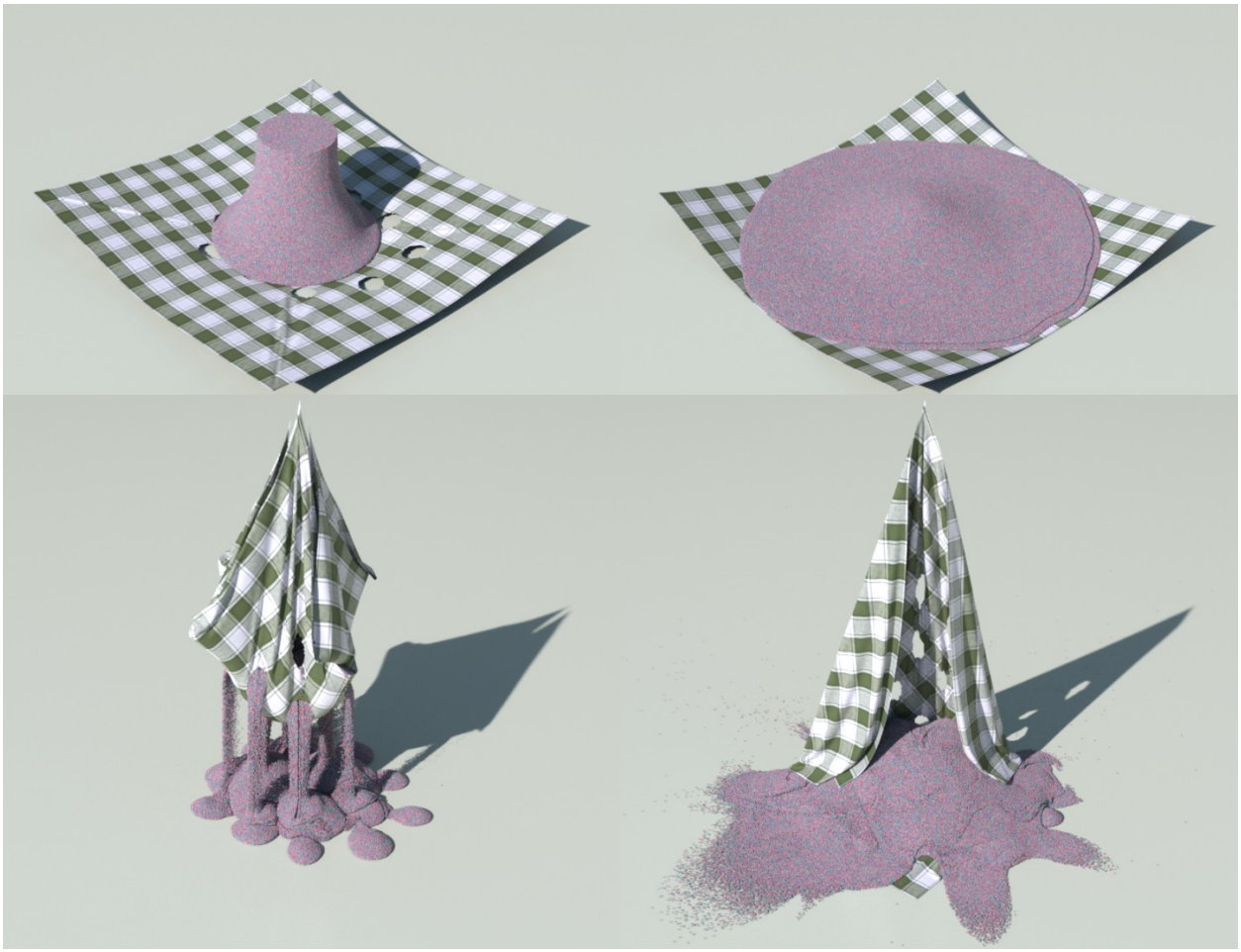


Figure 4.33: **Colored Sand.** A pile of MPM sand grains is scooped up by a piece of FEM cloth with holes.

Colored Sand In Figure 4.33, we showcase a simulation where a pile of colored sand, modeled using Drucker-Prager plasticity, is scooped up by a cloth. The sand grains slowly trickle out through small, pre-opened holes. Subsequently, the sand is released, burying a corner of the cloth. The rich collisions between the cloth and sand grains demonstrate the robustness of our coupling framework.

Debris Flow In our final example, depicted in Figure 4.34, we simulate a large-scale natural disaster involving a debris flow. The flow, modeled using von Mises plasticity with softening (Zhao et al., 2023b), cascades down a terrain of complex geometry. The tremendous kinetic energy of the debris flow not only causes significant deformation of trees but also washes rocks down the valley.

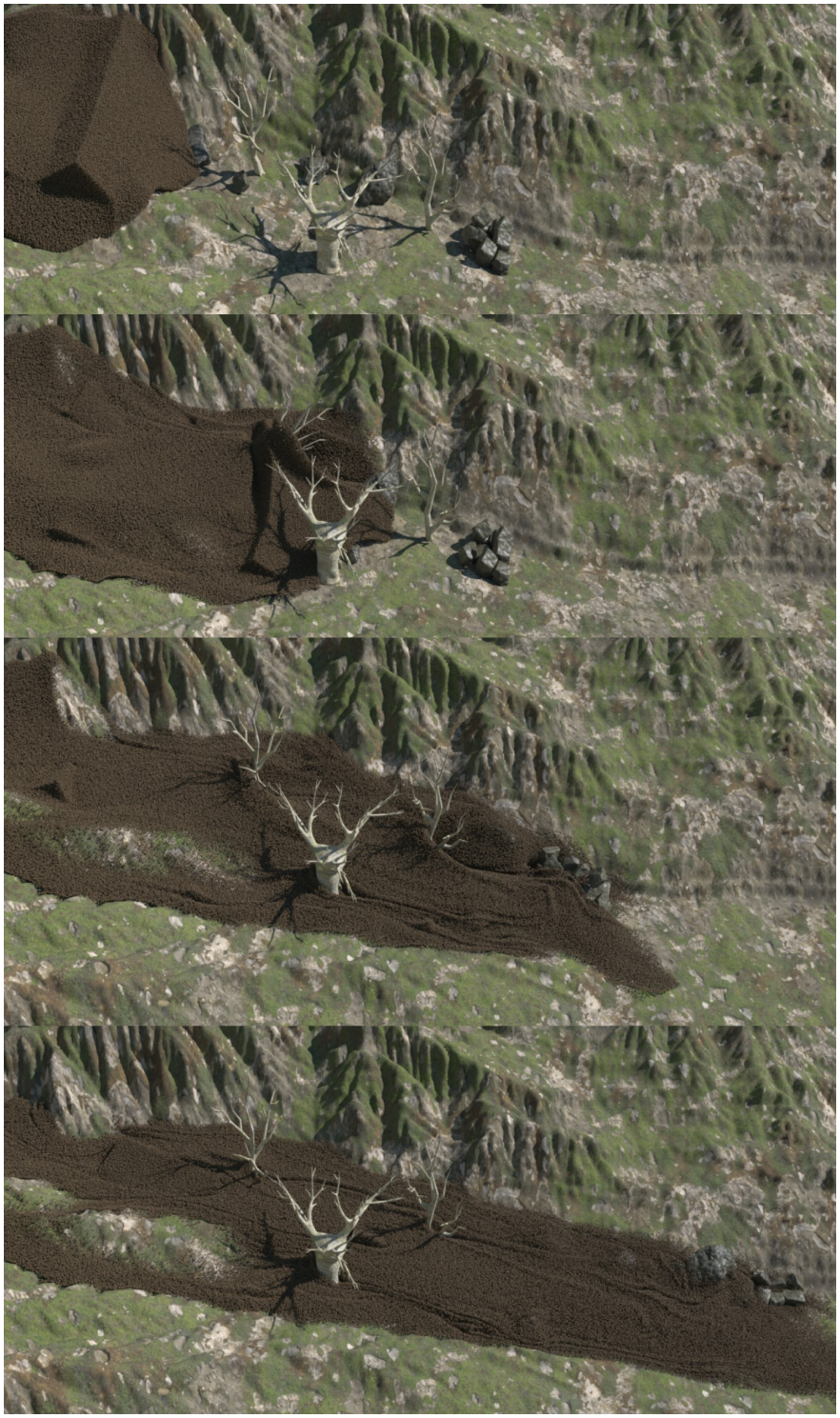


Figure 4.34: **Debris Flow.** A large-scale natural disaster caused by debris flow cascading down the valley.

4.2.6 Conclusion

We introduced the Dynamic Duo, a novel framework designed to integrate finite elements and material points seamlessly. The IMEX framework combines the optimal performance of implicit FEM and the flexibility of explicit MPM in applying various plasticity models. We achieve this through an asynchronous time-splitting scheme, where IPC is applied to model inter-domain frictional contact between FEM and MPM. The Dynamic Duo is not only pivotal for creating complex multi-material animations but also holds potential in inverse applications such as shape optimization, robot learning, and disaster prediction and prevention. However, our framework also presents certain limitations that warrant further research. For instance, friction clamping can underestimate the friction forces on MPM bodies. Tracking momentum loss and applying correction impulses to FEM could be explored. Moreover, completely eliminating the penetration issue in an efficient way is yet to be achieved. Future developments could explore alternating FEM and MPM integrations to incrementally resolve residual penetrations. Additionally, optimizing the coupling step is another avenue for improvement, possibly through more efficient Hessian assemblies and linear solvers.

CHAPTER 5

Towards Efficient Simulation

5.1 SPPD: Subspace-Preconditioned GPU Projective Dynamics with Contact for Cloth Simulation

5.1.1 Introduction

In cloth simulation, a fine and high-resolution discretization is often needed for rich and vivid effects like detailed wrinkles, folds, and creases, which coarser models cannot produce. The complexity, however, increases disproportionally w.r.t to the degrees of freedom (DOFs) of the system, making high-resolution simulation prohibitive for time-critical applications, whereas animating virtual garments at an interactive rate is always desired.

It is well understood that the primary obstacle for high-resolution cloth animation lies in the computational cost associated with the system solve for numerical integration at each time step. The cloth dynamics is nonlinear, and commonly used solvers rely on incremental linearization of the equation of motion e.g., see (Baraff and Witkin, 1998). Collisions and self-collisions, which are ubiquitous in cloth simulation, impose extra difficulties. Traditional methods often leverage soft repulsion to handle collisions (Tang et al., 2018; Wu et al., 2020). These approaches require careful parameter tuning to prevent undesirable artifacts like cloth interpenetration. The state-of-the-art solution to contact modeling is incremental potential contact (IPC) (Li et al., 2020), which utilizes log barriers and continuous collision detection (CCD) to strictly maintain objects' separation. The joint optimization with cloth elasticity is numerically challenging as the collision component is considerably stiffer and of higher frequency. The increased resolution also vastly complicates the spectrum of the system.

Figure 5.1: **Kick (High-Res).** Our method can efficiently simulate a high-resolution version (more than 120K nodes) of the Kick animation in CIPC (Li et al., 2021a) at 23s per frame, which is 6.5× faster than a heavily optimized and GPU accelerated CIPC solver. All collisions are robustly handled with intricate wrinkles captured on the cloth, highlighting the efficacy of our approach in handling fine-detailed garment simulations in complex animation scenarios.

While a wide range of numerical algorithms are available, they are proven only effective in limited or specific situations. For instance, gradient-based strategies (Wang and Yang, 2016) are quite parallelizable and efficient, but the performance declines quickly for stiffer instances. On the other hand, direct solvers are robust against numerical stiffness when paired with line search (Wolfe, 1969, 1971), but they are highly sequential and expensive for large-scale problems.

The pros and cons of existing algorithms endorse their strong complementarity, which forms our key rationale, illustrated in Figure 5.2. Specifically, we seek algorithmic synergy between direct and iterative numerical procedures for efficient high-resolution cloth animations. We argue that parallel local relaxation schemes e.g., Jacobi or Guass-Seidel are most effective if low-frequency residuals can be pre-eliminated. The latter happens to be the task in which subspace methods excel. The coordination of those two simulation modalities collectively delivers combined efficiency and convergence that existing methods can hardly match. Model reduction suppresses the system into a low-rank subspace, wherein the low-frequency residual can be solved efficiently. At the other end of the spectrum, the remaining high-frequency components become localized, making them an ideal target for parallel GPU solvers.

In this paper, we propose a novel subspace-preconditioned projective dynamics (PD) framework (Bouaziz et al., 2014) for cloth simulation. At each global step of PD, we solve for the displacement field within the subspace to capture low-frequency motion modes, with

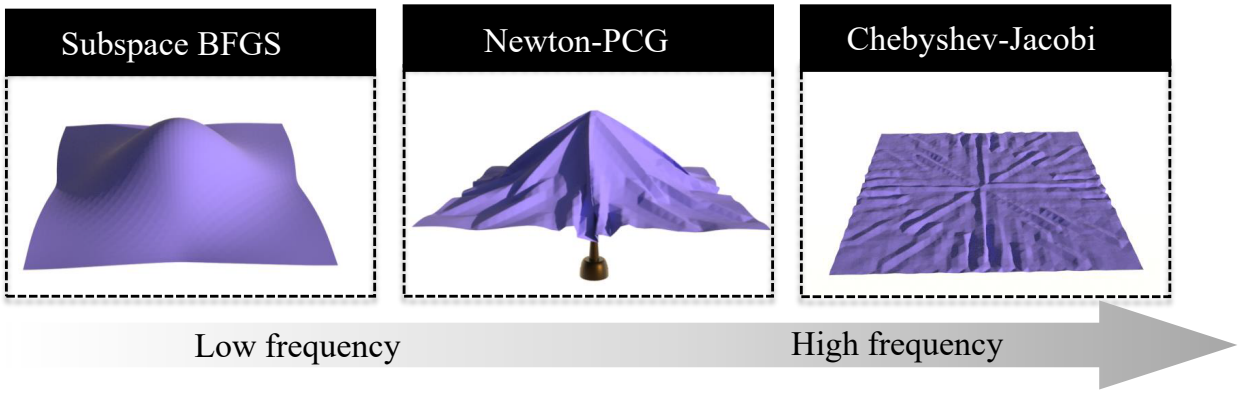


Figure 5.2: Subspace simulation, Newton method and Chebyshev-Jacobi excel at reducing residuals in different frequency ranges. Our method combines the advantages of Subspace BFGS and Chebyshev-Jacobi to achieve similar outcomes as Newton-PCG, but with improved performance.

a pre-factorized subspace global matrix. The high-frequency details are dealt with using parallel Chebyshev-Jacobi relaxation (Wang, 2015; Wathen, 2008) on the original full-order system. We adopt a time-splitting scheme along with a quadratic contact proxy (Xie et al., 2023a) to handle complex contacts. A novel modified Broyden-Fletcher-Goldfarb-Shanno (BFGS) method is incorporated to progressively integrate the quadratic contact proxy into the pre-factorized subspace global matrix, which circumvents the need for re-factorization. Our comparative evaluation demonstrates remarkable performance gain, enabling another 10× acceleration over the state-of-the-art simulation techniques – like IPC, the resulting animation is guaranteed to be free of any interpenetrations.

5.1.2 Related Work

Cloth Simulation Cloth and thin shell simulation, ubiquitous in daily life, continue to be central in the realm of computer graphics and animation (Grinspun et al., 2003; Gingold et al., 2004; Choi and Ko, 2005a). Modern cloth animation workflows largely incorporate an implicit time integration scheme, a practice pioneered by Baraff and Witkin (1998). Given the reduced stretchability of many fabrics, strain limiting is extensively used to prevent over-constraint (Provot et al., 1995; Goldenthal et al., 2007; Wang et al., 2010; Thomaszewski et al., 2009).

Considerable research has focused on refining material modeling to accurately emulate cloth's mechanical behaviors. For example, Volino et al. (2009) introduced a nonlinear and anisotropic tensile model based on continuum mechanics. Cloth bending is closely tied to the parameterization of the dihedral angle (Volino et al., 1995), with Breen et al. (1994) leveraging linear beam theory to link bending moment and curvature. Bridson et al. (2005) developed the bending mode of a hinge-based element orthogonal to in-plane deformations. In the context of inextensible fabrics, discrete mean curvature approximates bending, yielding a quadratic energy with a constant Hessian (Bergou et al., 2006; Wardetzky et al., 2007). The bending model introduced by Choi and Ko (2005b) effectively encapsulates compression and buckling. Furthermore, Kim (2020) unveiled the interconnections between the model proposed by Baraff and Witkin (1998) and classical Finite Element Method (FEM). Greater realism in cloth models can be achieved by incorporating captured real-world data (Wang et al., 2011; Feng et al., 2022; Miguel et al., 2012, 2013). Simulating garments at the yarn level, although computationally intensive, is feasible (Kaldor et al., 2008; Cirio et al., 2014), by condensing dynamics of individual yarn strands. Transitioning this approach onto triangle meshes results in much more tractable computations (Sperl et al., 2020).

Collision Handling Developing accurate contact models is crucial in mechanics, robotics, and computer graphics (Andrews et al., 2022; Johnson and Johnson, 1987). Traditional methods often handle contacts as constraint-based linear complementarity problems (LCP) (Kaufman et al., 2008; Baraff, 1994), resolved using projected Gauss-Seidel (PGS) method. An alternate approach uses quadratic programming (QP) (Redon et al., 2002; Macklin et al., 2020), compatible with more flexible solving techniques like projected gradient descent (Mazhar et al., 2015), mass splitting (Tonge et al., 2012), and the augmented Lagrangian method (Takahashi and Batty, 2021). Penalty methods (Bridson et al., 2002; Tang et al., 2012; Xu et al., 2014) are also employed to handle complex self-contacts. In the context of PD, Ly et al. (2020) introduced iterative refinement of contact forces during local steps.

While traditional methods mainly model contact through approximated constraints utilizing signed distances or volumes, recent approaches like the incremental potential contact

(IPC) model (Li et al., 2020) use precisely calculated unsigned distances for better robustness and accuracy. IPC approximates contact as a conservative force with a barrier potential, providing a controllable efficiency-accuracy tradeoff and ensuring penetration-free, convergent results for general contacts of codimensional solids (Li et al., 2021a), rigid bodies (Ferguson et al., 2021a; Lan et al., 2022a), hybrid multibody systems (Chen et al., 2022c), and FEM-SPH coupled domains (Xie et al., 2023a), etc. Despite its effectiveness, IPC's computational burden stems from the barrier function and the continuous collision detection in each nonlinear solver iteration. Recent endeavors have also concentrated on accelerating IPC through reduced-order models (Lan et al., 2021a), projective dynamics (Lan et al., 2022c), block coordinate descent (Lan et al., 2023), and time splitting (Wang et al., 2023b; Xie et al., 2023a), the majority of which also employ GPU acceleration.

Subspace Simulation Reduced-Order Modeling (ROM) offers a way to speedup the simulation of deformable bodies by using linear subspaces (Barbič and James, 2005; Sifakis and Barbic, 2012). These subspaces, usually built using modal analysis (Pentland and Williams, 1989; Hauser et al., 2003; Choi and Ko, 2005c) and its first-order derivatives (Barbič and James, 2005; Yang et al., 2015), simplify the model by removing less critical degrees of freedom (DOFs). This approach finds application in solids (An et al., 2008; Barbič and Zhao, 2011; Yang et al., 2015), fluids (Treuille et al., 2006; Kim and Delaney, 2013), and computational design problems (Xu et al., 2015). Alternative approaches include geometric shape coarsening, akin to skin rigging to prescribe the dynamics of a fine model. For instance, Capell et al. (2002) deformed an elastic body using an embedded skeleton, Gilles et al. (2011) employed 6-DOF rigid frames, Faure et al. (2011); Brandt et al. (2018) employed scattered handles, and Martin et al. (2010) used sparsely-distributed integrators for rods, shells, and solids.

Recent work has started to investigate nonlinear low-dimensional manifolds for ROM, with neural networks being used to construct these spaces (Lee and Carlberg, 2020). This approach can require smaller latent space dimensions compared to linear methods (Fulton et al., 2019; Shen et al., 2021). There has also been significant progress in data-driven latent

space dynamics (Lusch et al., 2018), with neural networks being used to learn the evolution of the entire latent space (Wiewel et al., 2019). To construct a subspace with sparse basis for general cloth dynamics, we stick with linear subspaces and use 2D B-spline functions as the building block.

5.1.3 Background

In this section, we provide a brief overview of the Projective Dynamics (PD) and Incremental Potential Contact (IPC) techniques, with a specific focus on cloth simulation, to ensure self-containment.

5.1.3.1 Projective Dynamics for Cloth Simulation

In the absence of collision, the PD solver employs the following optimization time integration for time stepping:

$$\min_{x} \frac{1}{2h^2}\|\boldsymbol{x}-\tilde{\boldsymbol{x}}\|_M^2 + \frac{E_{\text{mem}}}{2}\sum_t \|\boldsymbol{F}_t - \boldsymbol{R}(\boldsymbol{F}_t)\|^2 + \frac{E_{\text{bend}}}{2}\sum_e \|\boldsymbol{x}\|_{\boldsymbol{Q}_e}^2. \tag{5.1}$$

Here, h is the time step size, $\tilde{\boldsymbol{x}} = \boldsymbol{x}^* + \dot{\boldsymbol{x}}^* h + gh^2$ is the predictive position with $\boldsymbol{x}^*$ being the current position, $\boldsymbol{F}_t$ denotes the deformation gradient of triangle t, $\boldsymbol{R}(\boldsymbol{F})$ represents the closest rotation matrix to $\boldsymbol{F}$, E_{mem} and E_{bend} correspond to the membrane stiffness and bending stiffness, respectively, and $\boldsymbol{Q}_e$ is the local quadratic bending stiffness matrix for inner edge e, as outlined by Bergou et al. (2006).

Rather than directly optimizing the energy equation (5.1), PD decouples it by introducing auxiliary rotations $\boldsymbol{R}_e^k$ for each triangle, enabling optimization through a global-local alternating approach:

$$\begin{aligned} &\min_{\boldsymbol{x}^k} \frac{1}{2h^2}\|\boldsymbol{x}^k-\tilde{\boldsymbol{x}}\|_M^2 + \frac{E_{\text{mem}}}{2}\sum_t \|\boldsymbol{F}_t^k - \boldsymbol{R}_t^k\|^2 + \frac{E_{\text{bend}}}{2}\sum_e \|\boldsymbol{x}^k\|_{\boldsymbol{Q}_e}^2, \\ &\boldsymbol{R}_t^{k+1} = \boldsymbol{R}(\boldsymbol{F}_t^k). \end{aligned} \tag{5.2}$$

The global step involves solving a linear system with a fixed system matrix $\boldsymbol{H} = \frac{1}{h^2}\boldsymbol{M} + \boldsymbol{K}_{\text{mem}} + \boldsymbol{K}_{\text{bend}}$, where $\boldsymbol{K}_{\text{mem}}$, $\boldsymbol{K}_{\text{bend}}$ are membrane energy Hessian and bending energy Hessian at the rest shape. The local step can be executed efficiently in parallel. This alternating procedure continues until convergence is achieved. A precomputed Cholesky factorization can be applied to solve the global step. However, as the resolution increases, the computational time grows significantly. Moreover, multiple updates are required for the rotation matrices to ensure accuracy. To mitigate this, it is common to solve the global step inexactly using a few or even just one Jacobi iteration, while updating the rotation matrices as frequently as possible. To accelerate convergence, Chebyshev acceleration techniques can be applied, as suggested by Wang (2015). However, a substantial number of iterations are typically still required for high-resolution scenes.

5.1.3.2 Incremental Potential Contact

IPC (Li et al., 2020, 2021a) is an approach that handles contact constraints using smooth log barriers on unsigned distances to ensure separations between objects. It provides a robust method for processing collisions, where the log barrier potential is included as an additional energy term in the optimization-based time integration. By combining continuous contact detection (CCD), IPC can guarantee that there are no penetrations as long as the initial placement of objects is non-overlapping.

In the context of collisions between discrete meshes, collisions are classified into point-triangle and edge-edge collisions. The contact potential is defined as follows:

$$B(x) = \sum_{P,T} b(\text{dist}(P,T)) + \sum_{E_1,E_2} b(\text{dist}(E_1,E_2)), \tag{5.3}$$

where b is a smooth log barrier function of unsigned distance:

$$b(d) = \begin{cases} -(d-\hat{d})^2 \log(d/\hat{d}), & 0 < d < \hat{d}, \\ 0, & d \geq \hat{d}. \end{cases} \tag{5.4}$$

Algorithm 5 Timestepping of subspace-preconditioned PD

if it is the first time step **then**
  Construct subspace basis sparse matrix $\mathcal{P}$. ▷ Section 5.1.4.2
  Factorize the reduced-order global matrix $\mathcal{P}^\top \boldsymbol{H}\mathcal{P}$. ▷ Section 5.1.4.3
end if
Update predictive position $\tilde{\boldsymbol{x}}$.
Run a reduced-order global step w.o. contact for an initial guess.
Construct quadratic barrier proxy at current state $\boldsymbol{x}^*$.
Initialize subspace BFGS history.
while not converged **do** ▷ Section 5.1.4.5
  Run 2 iterations of subspace BFGS and update the history.
  Run 5 fullspace Jacobi iterations.
  Run PD local projections in parallel.
end while
Run penetration correction step. ▷ Section 5.1.4.6

Here, (P, T) represents an arbitrary point-triangle pair, and (E_1, E_2) represents an arbitrary edge-edge pair. The candidate pairs can be efficiently identified and filtered using bounding boxes. The parameter $\hat{d}$ controls the size of the contact zone. Inside this zone, the energy tends to infinity as the contact pair gets closer to each other, indicating the presence of contact forces. Outside the zone, there are no contact forces, and objects are considered separated. During the search for a energy-decreasing positional increment, CCD is employed to find an upper bound on the step size, such that when the step size is smaller, penetrations can always be avoided.

To incorporate frictional forces into the optimization, the IPC method integrates a locally smoothed semi-implicit Coulomb friction into a potential energy. For each contact point $\boldsymbol{x}_k$ with sliding basis $\boldsymbol{T}_k$ and normal contact force λ_k, the local friction is defined as

$$f_k(\boldsymbol{x}_k) = -\mu\lambda_k \boldsymbol{T}_k f_1(\|\boldsymbol{u}_k\|)\frac{\boldsymbol{u}_k}{\|\boldsymbol{u}_k\|}, \tag{5.5}$$

where μ is the frictional coefficient, $\boldsymbol{u}_k$ is the relative tangential velocity, f_1 is a function smoothly increase from 0 to 1 in the region $[0, \epsilon_v h]$ with ϵ_v controling the region of static friction.

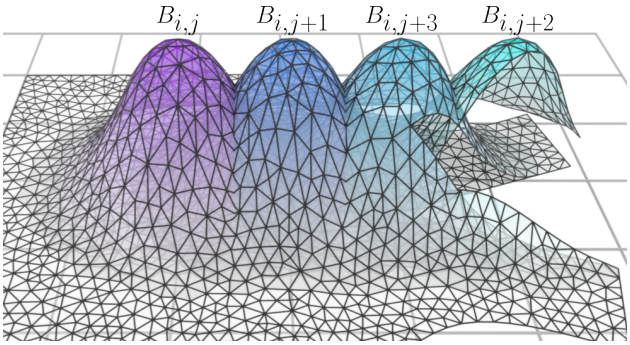


Figure 5.3: Basis functions defined in $\mathbb{R}^2$.

5.1.4 Method

5.1.4.1 Algorithm Overview

Our subspace preconditioned PD pipeline is summarized in Algorithm 5. We elaborate further details in the following subsections.

5.1.4.2 Construction of Subspace

Clothing items are typically composed of several flat fabric pieces, connected by stitches. Taking this into account, it is sufficient to utilize basis functions defined in $\mathbb{R}^2$. Suppose the cloth domain Ω is divided into multiple patches interconnected by stitches: $\Omega = \cup_{i=1}^{k} \Omega_k$. For each cloth patch Ω_k, we employ the As-Rigid-As-Possible (ARAP) parameterization technique (Liu et al., 2008) to obtain a bijective parameterization $\bar{\Omega}_k$. In certain cases, additional patch decompositions might be required to ensure bijectivity. Next, we embed each $\bar{\Omega}_k$ into a regular 2D Cartesian grid and employ Material Point Method (MPM) quadratic spline shape functions on the grid points (Jiang et al., 2016) as the basis for one spatial dimension of $\mathbb{R}^3$. Specifically, each basis is a product of two one-dimensional quadratic B-splines and is discretized on mesh nodes (see Figure 5.3):

$$B_{ij}(X) = N(u/\Delta x - i)N(v/\Delta x - j). \tag{5.6}$$

Here, (i, j) denotes a grid index, u and v represent the parameterization of X, Δx corresponds to the spline's kernel size and the spacing of the 2D grid, and $N(x)$ is defined as:

$$N(x) = \begin{cases} \frac{3}{4} - x^2, & |x| < \frac{1}{2}, \\ \frac{1}{2}(\frac{3}{2} - |x|)^2, & \frac{1}{2} \leq |x| < \frac{3}{2}, \\ 0, & \frac{3}{2} \leq |x|. \end{cases} \tag{5.7}$$

We decouple the three dimensions of the ambient space of Ω, meaning that the complete sparse basis matrix $\mathcal{P} = \mathcal{B} \otimes \mathcal{I}_3 \in \mathbb{R}^{3M \times 3N}$ is represented by the Kronecker product between the spline basis matrix $\mathcal{B} \in \mathbb{R}^{N \times M}$ for scalar functions and the 3D identity matrix, where M is the number of bases and N is the number of vertices.

Given a reference state of mesh position $\boldsymbol{x}_0$, we express the states in the subspace as $\{\boldsymbol{x} : \boldsymbol{x} = \boldsymbol{x}_0 + \mathcal{P}y, y \in \mathbb{R}^{3M}\}$. The displacement w.r.t. the reference state is constrained within the linear space expanded by the bases in $\mathcal{P}$. This basis satisfies the partition of unity property and is C^1-continuity w.r.t. the parameterization space.

Bending on Stitch If decompositions are necessary to achieve non-overlapping parameterization, such as for a tube-shaped cloth, we propose an approach to incorporate bending energy on the artificially generated stitches. As depicted in Figure 5.4, assume an artificial stitch passing through the shared edge of two triangles, namely (v_1, v_2, v_3) and (v_4, v_5, v_6). We introduce two bending energies, each with half of the original bending stiffness, on the 4-stencils (v_1, v_2, v_3, v_4) and (v_4, v_5, v_6, v_1). By doing so, the original energy is now split into two parts. As the stitch penalty pulls the vertices v_2 and v_5, as well as v_3 and v_6, closer together, the combined energy of these two parts will recover the energy prior to the decomposition.

5.1.4.3 Subspace Projective Dynamics

In each global step, as described in Equation 5.2, we need to solve a linear system $\boldsymbol{H}u = \boldsymbol{b}$, where u is the displacement increment w.r.t. the previous global step $\boldsymbol{x}^k$. To optimize the

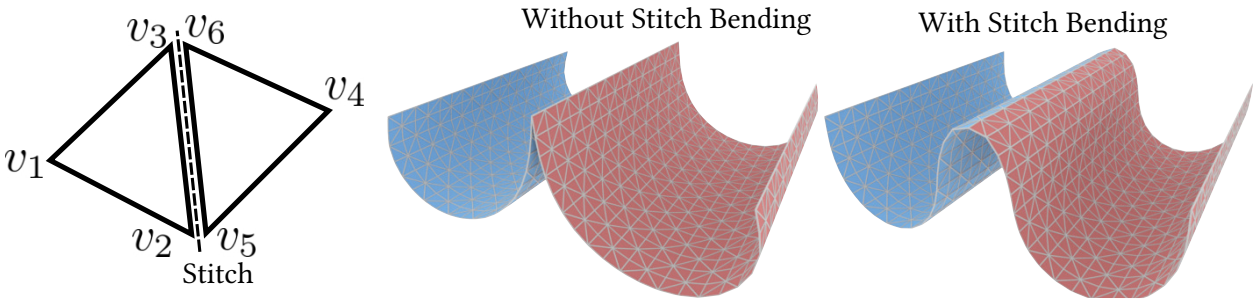


Figure 5.4: Stitch bending energies can recover bendings before domain decompositions.

quadratic energy within the subspace, we can restrict the displacement to the form $\boldsymbol{u} = \mathcal{P}y$. This leads to solving a reduced-order global system

$$\mathcal{P}^\top \boldsymbol{H} \mathcal{P} y = \mathcal{P}^\top \boldsymbol{b}. \tag{5.8}$$

Here, the reduced-order global system matrix $\hat{\boldsymbol{H}} = \mathcal{P}^\top \boldsymbol{H} \mathcal{P}$ has a dimension that corresponds to the number of bases in $\mathcal{P}$. This dimension can be much smaller than the total number of degrees of freedom. The advantage of this smaller matrix is that it can be prefactorized using the Cholesky decomposition, enabling efficient reuse for backsolving of Equation 5.8.

By exclusively solving the global step increment within the subspace, we effectively address the low-frequency modes that primarily govern the overall motion of the cloth. However, this approach tends to overlook the high-frequency details that showcase the benefits of high-resolution meshes. To reintroduce the high-frequency modes, we employ several Jacobi iterations on the original global system $\boldsymbol{H}\boldsymbol{u} = \boldsymbol{b}$, starting from the displacement obtained through the subspace solution. This two-level scheme bears a resemblance to a two-level multigrid method, wherein $\mathcal{P}$ and $\mathcal{P}^\top$ can be perceived as the restriction and prolongation matrices, respectively. Nevertheless, due to the necessity of a local projection step to update the membrane rotation reference, the right-hand side of the global linear system undergoes changes.

5.1.4.4 Subspace-Preconditioned Projective Dynamics with Contact

We follow the same strategy of Xie et al. (2023a) and incorporate contact in a specifically designed time-splitting manner. At the beginning of each time step, we introduce a quadratic proxy of the contact potential into the PD solver, allowing for penetration. Subsequently, an additional correction step is incorporated to fix all penetrations while minimizing changes as much as possible.

5.1.4.5 PD with Quadratic Contact Proxy

The quadratic proxy is defined as the second-order expansion of the barrier potential at the initial state $\boldsymbol{x}^*$ of the current time step:

$$\hat{B}(\boldsymbol{x};\boldsymbol{x}^*) = B(\boldsymbol{x}) + \nabla B(\boldsymbol{x}^*)^\top(\boldsymbol{x}-\boldsymbol{x}^*) + \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{x}^*\|^2_{\nabla^2 B(\boldsymbol{x}^*)}. \tag{5.9}$$

By combining this quadratic proxy, the global system of the PD solver is still linear:

$$(\boldsymbol{H} + \nabla^2 B(\boldsymbol{x}^*))\boldsymbol{u} = \boldsymbol{b} - \nabla B(\boldsymbol{x}^*) - \nabla^2 B(\boldsymbol{x}^*)(\boldsymbol{u} + \boldsymbol{x}^k - \boldsymbol{x}^*), \tag{5.10}$$

where $\boldsymbol{H}\boldsymbol{u} = \boldsymbol{b}$ represents the global step system without contact.

However, the global system matrix is subject to change over time. To reuse the prefactorized matrix obtained without contact, we employ subspace BFGS iterations to minimize the quadratic energy of the global step within the subspace, i.e., we solve the quadratic problem with the increment constrained to the subspace. It is important to note that the subspace global matrix is now:

$$\mathcal{P}^\top(\boldsymbol{H} + \nabla^2 B(\boldsymbol{x}^*))\mathcal{P} = \mathcal{P}^\top \boldsymbol{H}\mathcal{P}^\top + \mathcal{P}^\top \nabla^2 B(\boldsymbol{x}^*)\mathcal{P}^\top. \tag{5.11}$$

The prefactorized matrix $\mathcal{P}^\top \boldsymbol{H}\mathcal{P}$ can serve as the initial approximation of the Hessian for the subspace BFGS iterations at the beginning of each time step. Importantly, the global system matrix remains unchanged within a single time step, allowing for the reuse of BFGS

history across different global steps within the time step. To ensure efficiency, we limit the number of BFGS iterations to 2 in each global step, effectively providing only one additional 2-rank update to the initial reduced-order Hessian matrix based on previous updates. For quadratic problems, the optimal step size for line search can be computed analytically, requiring only one matrix-vector multiplication. It is worth mentioning that at the begining of the global-local alternations, we solve one reduced-order global step, where one backsolve using the prefactorized subspace global system can give us the exact solution. This initial guess significantly decreases the number of PD iterations.

Following the subspace BFGS iterations, we perform 5 block-diagonal Jacobi iterations on the original full-order linear system (Equation (5.10)) to enrich high-frequency details in the solution. This choice aligns with the common practice of multigrid, which incorporates 3–5 smoothing iterations per cycle. The block size is 3 because all matrices are assembled with 3×3 blocks, during which the diagonal blocks are tracked. However, with contact proxy stiffness matrix, the eigenvalue of the iteration matrix can easily exceed 1. Here we use a modified Jacobi with automatically tuned weight. We observe that each naive Jacobi iteration is essentially a block-diagonal preconditioned gradient descent for the corresponding quadratic problem. The steepest descent step size can be analytically computed as discussed above. We use that step size as the weight for each Jacobi iterations. This can make sure that the energy for the global step in Equation 5.2 is always decreasing.

In summary, each global step of our solver consists of 2 L-BFGS iterations and 5 Jacobi iterations. The PD phase is terminated if the L-infinity distance between states from two consecutive global steps falls below a given tolerance ($5 \times 10^{-3}h$ in our expriments with time step size h), or if the maximum number of iterations is reached (200 in our experiments). Additionally, we apply Chebyshev acceleration to the global steps, following Wang (2015); Liu et al. (2017b), to accelerate convergence. Empirically, we found that Chebyshev weight 0.99 worked well across all our examples.

5.1.4.6 Penetration Correction

The above PD solver may provide a trial solution x^{tr} with penetrations. To resolve these penetrations, we solve the following energy minimization using projected Newton method combined with a non-penetration line search, following the approach used in the IPC framework:

$$\frac{1}{2h^2}\|x - x^{\text{tr}}\|_M^2 + B(x). \tag{5.12}$$

This objective function aims to find a balance between the trial state from PD and the collision constraints. Unlike the original IPC, which solves the linear system in the Newton method with Cholesky factorization, we employ a block-diagonal preconditioned conjugate gradient (PCG) method to solve the system. On GPUs, iterative solvers are generally much faster than direct solvers. Furthermore, we use a smaller contact stiffness than the quadratic proxy. The stiffness for the proxy is slightly lower than the elasticity stiffness, while in penetration correction, the value is adjusted to $0.01\times$. This adjustment can reduce the condition number of the nonlinear optimization problem, facilitating faster convergence.

We also propose an early-stop strategy to reduce the number of Newton iterations required in this step. The objective of this phase is to resolve collisions while preserving momentum as much as possible. By observing this, we can increase the tolerance for DOFs that are involved in contacts, while focusing on preserving momentum primarily for DOFs that are not in contact. By prioritizing momentum preservation for non-contact DOFs and allowing for a slightly larger tolerance for contact DOFs, we can reduce the number of Newton iterations required in the penetration resolution step while still achieving satisfactory results. In our experiments, the tolerances on non-contact DOFs and contact DOFs are $10^{-2}h$ and $10^{-1}h$ respectively.

5.1.4.7 Friction

We also introduce an approach designed specifically for our time-splitting contact model incorporating frictional effects. This model utilizes the Hessian of the frictional energy as a damping matrix. We note that the friction coefficient μ no longer holds a physical meaning

but still controls the magnitude of the frictional forces. In this friction proxy, we expand the frictional contact potential at $x_k = x_k^* + 10h\epsilon_v \dot{x}_k^* / \|\dot{x}_k^*\|$, and remove the contributions of tangential velocities smaller than ϵ_v. That is, we only adopt dynamic friction Hessians for damping along the tangential directions. Our fuzzy friction proxy can be seamlessly integrated into the contact proxy and incorporated into the PD solve loops using the BFGS algorithm.

5.1.5 GPU Implementation

Our algorithm has been implemented to run efficiently on a single GPU using CUDA 12.1. To avoid write-write conflicts during the assembly of global gradients and Hessian matrices, we did not utilize coloring algorithms like the one proposed in Fratarcangeli et al. (2016). Instead, we found that using atomic add operations on our GPU was already efficient enough and simpler to implement. For matrix-vector products in the Jacobi solver, CG method, and subspace restriction/prolongation, we employed cuSPARSE, a library for efficiently performing operations on compressed sparse row (CSR) matrices in CUDA. In addition, the dense Cholesky factorization of $\mathcal{P}^\top \boldsymbol{H} \mathcal{P}$ was implemented using cuSOLVER, which enabled efficient factorization of the reduced-order global matrix. At each Newton iteration during the penetration correction, CCD is required on each search direction to guarantee non-penetration. We use the patch-based GPU collision culling from Lan et al. (2022c) to efficiently reduce the number of candidates.

5.1.6 Experiment

We implemented our algorithm on a desktop workstation with an NVIDIA RTX 3090 GPU and an Intel Core i9-10920X 3.5-GHz CPU with 12 cores. We also follow (Macklin et al., 2019) using simulation substeps for optimized performance.

Figure 5.5: The comparison between our method and Wang (2015) on a cloth hanging experiment. The cloth used in the experiment consists of 250K vertices and 500K triangles. Our method demonstrates closer agreement with the results obtained using Newton's method, but is much faster.

5.1.6.1 Compare with Chebyshev-Accelerated Jacobi-PD

We compare our method with (Wang, 2015), a classic GPU-accelerated PD algorithm using Jacobi method to solve the global system inexactly. In this test, we simulate a piece of table cloth (250K vertices, 500K triangles) with two upper corners fixed. Here, we exclude collision and self-collision processing in both methods to only showcase the performance-wise difference. For the method described in Wang (2015), we performed three separate experiments with 1,000, 2,000, and 3,000 Jacobi iterations, respectively. We also simulate the scene using Newton's method as the reference. The results of the comparison are illustrated in Figure 5.5. Even with 3,000 iterations, Wang (2015) exhibits more discrepancies with Newton's results compared to our method. Furthermore, our method demonstrates faster computation times compared to Wang (2015) when utilizing only 1000 iterations, despite the presence of artifacts in their results.

5.1.6.2 Compare with Hyper-Reduced Projective Dynamics

Our work differs from hyper-reduced PD (HRPD) (Brandt et al., 2018) in several key aspects. The subspace design in our method is tailored for cloth and shell structures, whereas HRPD targets volumetric solids. Additionally, we solve the full-order system, while HRPD directly

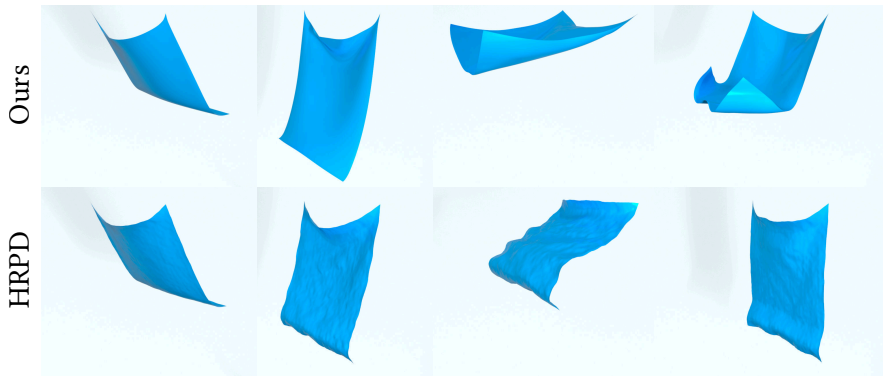


Figure 5.6: Our B-spline bases on regular grids naturally satisfy the property of partition of unity, while hyper-reduced bases (Brandt et al., 2018) do not, leading to severe locking under large deformations.

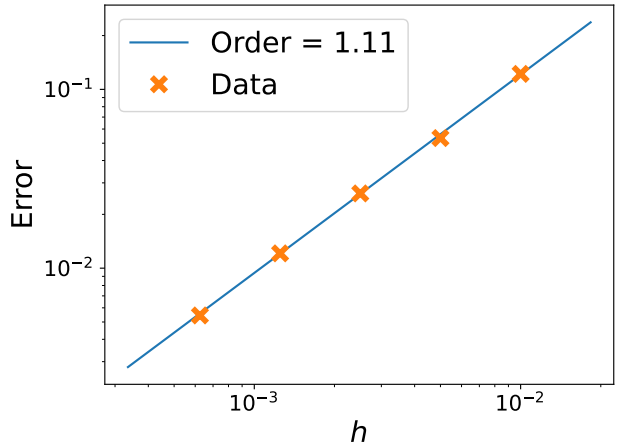


Figure 5.7: Convergence under refinement.

simulates within the reduced subspace. Furthermore, the B-spline bases in our approach inherently satisfy partition of unity on regular grids. This is advantageous for simulating cloth, as HRPD bases lack this property. To demonstrate, we compare HRPD to our method by directly simulating the subspace dynamics without Jacobi relaxation (ours used 1200 bases, while HRPD used 1497 bases). As shown in Figure 5.6, the hyper-reduced subspace exhibits severe locking under large deformations, while our method does not.

5.1.6.3 Ablation Study

Convergence Under Refinement To evaluate the accuracy of our cloth solver, we perform a convergence under refinement test w.r.t. the time step size h. We begin by capturing a

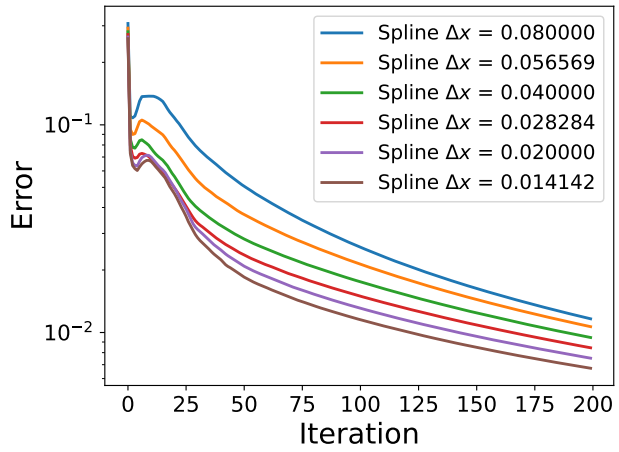


Figure 5.8: Ablation study on the number of spline bases.

contact-rich snapshot of the "cloth on sphere" experiment and using it as the initial state. We then simulate with CIPC using h=1e-4 for a duration of 0.1 seconds. The final state obtained from the CIPC simulation serves as the reference. We then use our cloth solver to simulate with consecutively halved time steps, starting from $h = 0.01s$, in order to estimate the convergence order of the L^2-error to the reference. Shown in Figure 5.7, the estimated convergence order is 1.11.

Number of Spline Bases The rate of convergence in the PD phase depends on the number of spline bases. In our analysis, we take a contact-rich snapshot and focus on a single substep. The outcome from the CIPC solver serves as the reference state, against which we evaluate the convergence of our BFGS-based PD solver across varying numbers of spline bases. In our framework, the number of spline bases is controlled by the spacing between spline centers. As depicted in Figure 5.8, a smaller spline spacing Δx typically leads to less iterations for convergence. However, extremely small Δx values can result in too much computational time of the backsolve step on GPU. We estimated that this component's timing is approximately $O(\frac{1}{\Delta x}^{1.9})$. Despite efforts, we haven't identified an optimal empirical compromise between timing and PD convergence speed. Nevertheless, maintaining the number of bases slightly below 10K for a 100K-vertex cloth tends to yield favorable overall speedups.

Table 5.1: Average computational cost per frame (s) in the comparisons.

Experiment	#V	$\hat{d}$ (m)	#Basis	#Sub-step	Ours	PD Step Percentage	GPU CIPC	CIPC	PD-IPC
Ribbons	285K	2e-4	7119	16	**46**	43.3%	1033	1579	134
Cloth on sphere	252K	1e-4	8427	8	**27**	37.2%	241	2040	49
Funnel	232K	5e-4	9432	10	**23**	63.3%	362	2740	253
Reef knot	104K	3e-4	7665	8	**10**	81.9%	90	144	18

5.1.6.4 Benchmarks

We further compare our method with two known IPC-based cloth simulation methods, namely CIPC (Li et al., 2021a) and PD-IPC (Lan et al., 2023), in multiple high-resolution cloth simulation setups. The original CIPC implementation was on the CPU, which is quite expensive. To avoid misleading benchmarks from different platforms, we re-implemented CIPC on the GPU, and we refer to our own implementation as GPU CIPC. GPU CIPC port most costly computations to CUDA including collision detection, culling, CCD, and Newton solve, and it already exhibits notable performance gains compared to its CPU counterpart. Nevertheless, our method (proposed in this paper) offers further speedups. PD-IPC is our closest competitor as it is also based on the PD. However, they utilize simplified formulations for the membrane and bending energies compared to our method. We made best efforts to visually match their results with ours under the same numerical settings.

In the comparative analysis, we used a frame duration of 0.04s. We try our best to tune the time step size to ensure a fair comparison of performance among different methods. In the case of CPU IPC, the most efficient time step size is typically the frame duration. This is because the direct solver used in CPU IPC is not sensitive to the condition numbers of the Hessian matrix. However, for most iterative methods, including ours, substepping is beneficial they are more dependent on the condition numbers of the problem. The per-frame computational costs of different approaches are listed in Table 5.1.

Ribbons (Figure 5.9) We simulate the behavior of 21 long ribbons dropped into a bowl. The ribbons interact with each other, resulting in multiple collisions and self-collisions. In this example, we achieve 22× acceleration compared to GPU CIPC, 34× acceleration compared

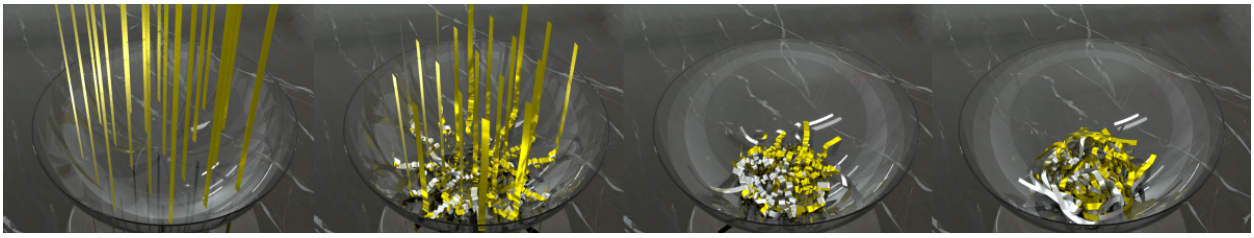


Figure 5.9: **Ribbons**. Twenty-one long ribbons are dropped into a round bowl, leading to numerous collisions and self-collisions among the ribbons.

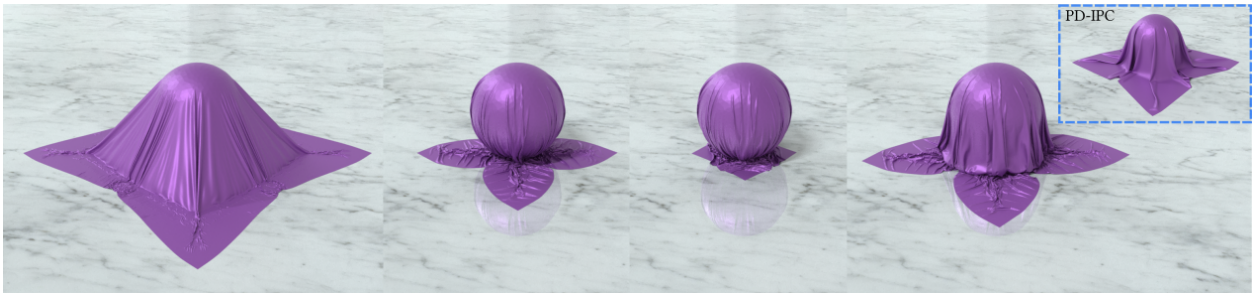


Figure 5.10: **Cloth on sphere**. A square cloth is dropped onto a sphere and form numerous intricate wrinkles. PD-IPC utilizes simplified strain and bending models, resulting in inability to capture fine-detailed wrinkles.

to CPU CIPC, and 2.9× acceleration compared to PD-IPC. We note that PD-IPC utilizes simplified strain and bending models. In this example, it is not able to capture fine-detailed wrinkles, resulting in fewer contact interactions compared to our method.

Cloth on sphere (Figure 5.10) A square cloth is dropped onto a sphere, with a ground surface located beneath the sphere. There are persistent contact between the center of cloth with the top of shpere, and persistent contact between the cloth and the ground. The contact results in numerous intricate wrinkles. We achieve 8.9× acceleration compared to GPU CIPC, 75× acceleration compared to CPU CIPC and 1.8× acceleration with PD-IPC.

Funnel (Figure 5.11) To evaluate the robustness of our contact handling, we conduct a test involving three pieces of cloth dropped onto a shallow funnel. As a scripted sphere passed through the hole of the funnel at a constant speed, multiple collisions occurred between the layers of cloth and between the cloth and the funnel. This scenario presented a significant challenge due to the substantial compression experienced by the cloth while passing through

Figure 5.11: **Funnel**. Three pieces of cloth are dropped onto a shallow funnel. A sphere pushes the layers of cloth completely through the hole of the funnel.

Figure 5.12: **Reef knot.** Two curved ribbons are initially intertwined and then pulled in opposite directions to form a tiny and tight knot.

the hole. In this example, our method exhibited remarkable performance gains compared to other approaches. Specifically, we achieved a 15× acceleration compared to GPU IPC, a 119× acceleration compared to CPU IPC, and an 11× acceleration compared to PD-IPC. These results highlight the robustness and efficiency of our contact handling technique in challenging cloth simulations.

Reef knot (Figure 5.12) Two curved ribbons are initially intertwined and then pulled in opposite directions to form a knot. It is worth mentioning that, to apply our method, each ribbon is decomposed into three pieces combined with stitch bendings, as depicted in Figure 5.4. Our method manages to generate a tiny and tight knot. And we achieve 9× accelerateion comapred to GPU-IPC, 14× acceleration compared to CPU IPC, and 1.8× acceleration compared to PD-IPC.

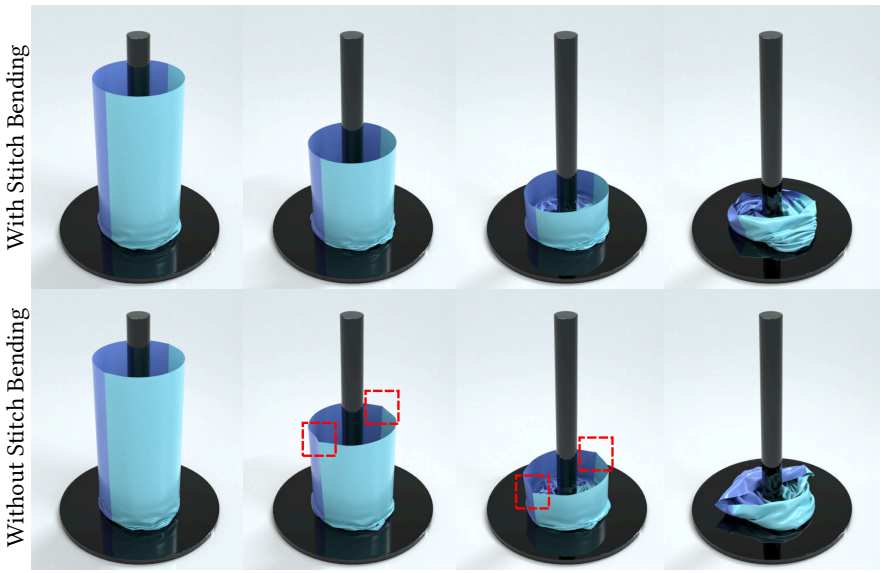


Figure 5.13: Stitch bending is critical for preserving the correct bending stiffness along the artificially generated seam, which does not exist on the original cloth surface.

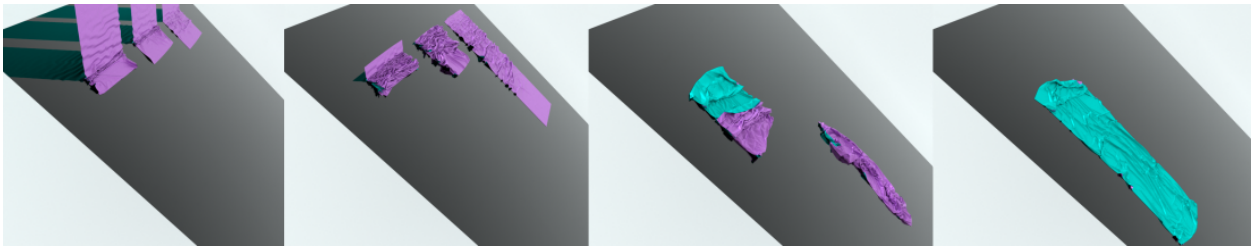


Figure 5.14: **Controllable friction.** Three pieces of 120K-node long rectangle cloth fall onto the slope under gravity. With larger friction coefficients (closer to the camera), the cloth slides down the slope at a slower rate until it gets stuck and then undergoes turning motions.

5.1.6.5 Example with Stitch Bending

Some shapes needs artificial decompositions to obtain 2D ARAP parameterizations, such as a cylinder. Here we simulated a cylindrical cloth mesh, which is decomposed into two pieces. Stitch bending is critical for preserving the correct bending stiffness along the artificially generated seam, which does not exist on the original cloth surface. Without stitch bending, the cloth exhibits an unrealistically large fold along the seam, as shown in Figure 5.13.

5.1.6.6 Controllable Friction

In Figure 5.14, we present an experiment where we vary the friction coefficients between the cloth and the slope. The cloth used in the experiment is a long rectangle with 120K vertices, and it falls onto the slope under gravity. By increasing the friction coefficient, we observe that the cloth slides down the slope at a slower speed until it gets stuck and then undergoes turning motions.

5.1.6.7 Garment Animation

Fine-detailed garment simulations play a crucial role in the animation industry. In the animation pipeline, artist-designed character animation sequences serve as moving boundary conditions for the cloth simulations. However, simulating garments on animated characters presents significant challenges due to the dramatic motions involved, such as running, jumping, or dancing. Here we use a challenging test case from Li et al. (2021a), where a character turns and kicks wearing a multi-layer dress (Figure 5.1). We subdivided the original testing garments to 120K vertices. The leg causes the dress moving in a high speed and there are intricate interactions between layers of the dress. Our method can robustly handle these collisions and resolve complex wrinkles on the cloth, with an average running time of 23 seconds per frame. This is a 6.5× acceleration compared to the GPU IPC method, highlighting the effectiveness and efficiency of our approach in handling fine-detailed garment simulations in complex animation scenarios.

5.1.7 Conclusion

In this paper, we propose an efficient cloth simulation method based on the projective dynamics (PD) framework. Our method combines subspace integration and parallelizable iterative relaxation techniques to effectively reduce both high-frequency and low-frequency residuals, leading to significantly improved convergence. We seamlessly integrate our method with the state-of-the-art contact handling framework, IPC, to ensure interpenetration-free results in a time-splitting manner. We have shown that our method has significant performance

improvements over existing GPU solvers for high-resolution cloth simulation.

Indeed, when dealing with objects exhibiting high speeds, the time splitting error can become significant, leading to amplified damping effects. To address this issue, it would be valuable to explore adaptive substepping techniques that can enhance the accuracy of the time splitting process and alleviate the undesired damping artifacts.

Furthermore, the use of Newton's method in the penetration correction step may give rise to overshooting problems, resulting in excessive optimization iterations. Notably, the penetration correction step typically consumes a substantial portion of the computation time. To further improve the overall performance of our algorithm, it is crucial to investigate dedicated solvers specifically tailored for the penetration correction step.

5.2 XPBI: Position-Based Dynamics with Smoothing Kernels Handles Continuum Inelasticity

5.2.1 Introduction

Position-based Dynamics (PBD) (Müller et al., 2007) and its extension eXtended Position-based Dynamics (XPBD) (Macklin et al., 2016) are widely adopted in compliant constrained dynamics, particularly favored for their performance and simplicity for graphics applications such as rigid bodies (Müller et al., 2020), soft bodies (Bender et al., 2014), cloth (Müller et al., 2007), rods (Umetani et al., 2015) and hair (Müller et al., 2012). When simulating mesh-based elasticity, it is straightforward to model XPBD constraints with FEM hyperelastic energies defined over explicit mesh topology. This allows for effective simulations of elasticity while maintaining the stability and speed of XPBD. For example, quadratic energy potentials (Chen et al., 2023c) can be formulated using XPBD-style constraints. Macklin and Müller (2021) and Ton-That et al. (2023) reformulated stable Neo-Hookean (Smith et al., 2018) to demonstrate XPBD's capability in handling nonlinear elasticity.

For inelasticity, on the other hand, two significant challenges emerge. Firstly, topology changes during material splitting and merging introduce great complexity in maintaining a

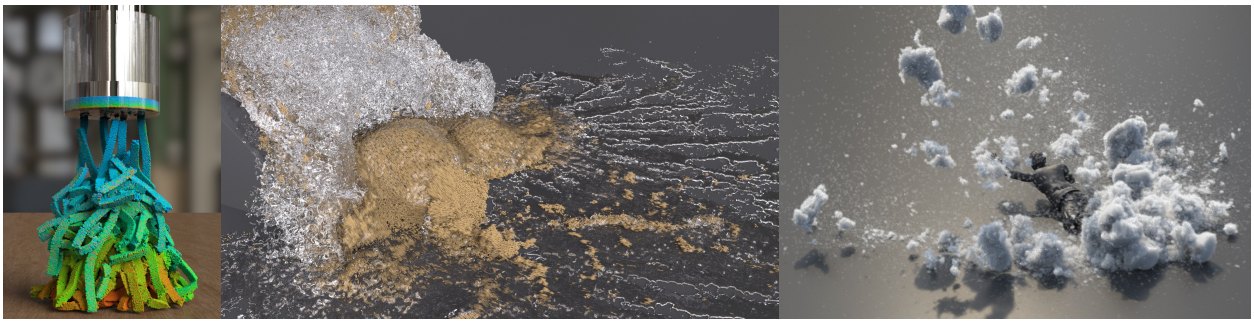


Figure 5.15: XPBI supports simulating a wide range of classical continuum elastoplastic material models such as Von-Mises plasticine, Drucker-Prager sand, and Cam Clay snow, as well as their interactions with traditional PBD materials such as Position-based Fluid.

high quality mesh, often necessitating remeshing. Secondly, while there have been explorations that enhances Position-based Fluids (PBF) (Macklin and Müller, 2013) with the conformation tensor (Barreiro et al., 2017) for viscoelastic fluids, it remains underexplored for XPBD to model physically-grounded finite strain (visco-) elastoplastic constitutive laws from classical continuum mechanics, such as von-Mises (Mises, 1913b), Drucker-Prager (Drucker and Prager, 1952) and Herschel-Bulkley (Herschel and Bulkley, 1926) flow rules. Being able to simulate them would greatly improve XPBD's versatility and intuitive controllability of material parameters.

In comparison with XPBD's success in mesh-based materials, Material Point Method (MPM)'s development in graphics over the past decade has majorly focused on inelastic phenomena with topology change. MPM is based on the weak form of governing PDEs and employs a hybrid Lagrangian-Eulerian approach for spatial discretization. It handles topology changes, self collision, and finite strain deformation without overhead, and thus has been used for many continuum inelastic phenomena such as snow (Stomakhin et al., 2013), lava (Stomakhin et al., 2014), sand (Klár et al., 2016), mud (Tampubolon et al., 2017), metal (Wang et al., 2020a), foam (Ram et al., 2015), and fracture (Wolper et al., 2019). While substantial progress has been made in these areas, MPM exhibits several notable drawbacks, including excessive numerical dissipation due to particle-grid transfers (Jiang et al., 2015a), artificial stickiness that hampers material separation (Fei et al., 2021a), and resolution-dependent gaps between colliding materials (Jiang et al., 2017a). None of these

Table 5.2: XPBI keeps PBD's pure particle nature of the Degrees of Freedom while allowing MPM-style plasticity and granular material modeling through an updated Lagrangian treatment of the deformation and classical continuum mechanics-based elastoplastic flow rules.

	Lagrangian	**DOF**	**Plasticity**	**Granular**
MPM	Updated	Grid	Flow Rule	Continuum
PBD	Total	Particle	N/A	Sphere Approx.
Ours	Total & Updated	Particle	Flow Rule	Continuum

issues are present in XPBD. This raises a natural question: *Can we simulate MPM-style phenomena using XPBD instead?*

Towards addressing this question, we make an important observation: the primary factor that facilitates the modeling of inelasticity lies not in the hybrid Lagrangian-Eulerian nature of MPM, but rather in its use of an updated Lagrangian formulation for the deformation gradient tensor. In particular, one considers the time $n+1$ velocity $\boldsymbol{v}^{n+1}$ to be defined over the previous time n domain Ω^n through the Lagrangian velocity $\boldsymbol{V}$ of a particle traced back using the inverse deformation map $\boldsymbol{\phi}^{-1}(\boldsymbol{x}, t)$: $\boldsymbol{v}^{n+1}(\boldsymbol{x}) = \boldsymbol{V}(\boldsymbol{\phi}^{-1}(\boldsymbol{x}, t^n), t^{n+1})$ (Jiang et al., 2016), where t represents the continuous time variable. This enables the derivation of the rate form of the deformation gradient $\boldsymbol{F}$ given by $\dot{\boldsymbol{F}} = (\nabla \boldsymbol{v})\boldsymbol{F}$, which can be further discretized into $\boldsymbol{F}^{n+1} = (\boldsymbol{I} + \Delta t \nabla \boldsymbol{v}^{n+1})\boldsymbol{F}^n$, where Δt is the time step size, allowing one to track the deformation gradient *without referring to a material space configuration.*

Inspired by this observation, if we can track the deformation gradient tensor $\boldsymbol{F}$ using an updated Lagrangian view in XPBD, then by treating $\boldsymbol{F}$ as a function of XPBD degrees of freedom, we can modify XPBD to resemble a "material point" approach. As detailed in later sections, this task reduces to robustly computing and differentiating the velocity gradient tensor. We present developments surrounding these ideas by introducing eXtended Position-based Inelasticity (XPBI), where the X represents not only the incorporation of XPBD augmentation but also the use of velocity as the primal variable. This approach computes the updated Lagrangian deformation gradient using a velocity-based formulation, allowing us to handle various inelastic effects. Using velocities as primary variables allows for direct evaluation of the velocity gradient and particle-wise constraints using interpolation kernels defined at t^n, aligning with standard MPM practices, while using positions could introduce

uncertainties in updating kernels during implicit iterations. By further incorporating an implicit plasticity treatment and additional stability-enhancing components, our method leverages the efficiency and simplicity of PBD while capturing the complex inelastic material responses typically associated with MPM; see Table 5.2. In summary, our contributions include:

- An updated Lagrangian augmentation for XPBD that tracks meshless deformation gradients and per-particle constraints;
- XPBI, a fully implicit plasticity-aware algorithm capable of handling continuum mechanics-based elastoplastic/viscoplastic laws;
- An investigation for practical stability enhancements, such as XSPH and position correction, and validations of our method with various practical examples.

5.2.2 Related Work

Inelasticity with PBD Müller et al. (2007) introduced PBD, which replaces internal forces with positional constraints and produces appealing, stable and real-time simulations. Its first-order convergence was studied by Plunder and Merino-Aceituno (2023). XPBD (Macklin et al., 2016), an extension of PBD, utilizes the compliant-constraint framework (Tournier et al., 2015) to uniformly handle soft and hard constraints to simulate elasticity. Our work follows the latest XPBD paradigm (Macklin et al., 2019) with substeps. Other PBD materials include rigid body (Müller et al., 2020), soft body (Bender et al., 2014), cloth (Müller et al., 2007), hair (Müller et al., 2012), elastic rod (Umetani et al., 2015), sand (Macklin et al., 2014), fluid (Macklin and Müller, 2013) with surface tension (Xing et al., 2022) and their unified couplings (Macklin et al., 2014; Frâncu and Moldoveanu, 2017; Abu Rumman et al., 2020). We refer to Bender et al. (2017) for a comprehensive survey.

For *continuum* materials, Bender et al. (2014) defined a constraint for the elastic strain energy. Müller et al. (2015) constrained the strain tensor directly instead. Macklin and Müller (2021) reformulated stable Neo-Hookean using XPBD. Plastic deformation and fracture can

be modeled by shape matching (Chentanez et al., 2016; Jones et al., 2016a; Falkenstein et al., 2017), prioritizing efficiency over accuracy. Macklin et al. (2014) simulated sand as colliding spheres with friction. SideFX Houdini's Vellum PBD solver further added spring-like cohesion for snow. Without further utilizing continuum mechanics-based inelastic models, these approaches have limited mechanical intuition and physical parameter controllability. A step forward was proposed by Barreiro et al. (2017), which enhances PBF with the conformation tensor for viscoelastic fluids. Nevertheless, it does not incorporate finite strain continuum mechanics, limiting its suitability in modeling general elastoplastic and viscoplastic laws.

Inelasticity with MPM MPM was introduced by Sulsky et al. (1994) as a hybrid Lagrangian/Eulerian approach for solids. Since its adoption in graphics (Hegemann et al., 2013; Stomakhin et al., 2013), MPM has gained significant attentions for its automatic topology change and material versatility. Snow plasticity was first done by projecting principal stretches (Stomakhin et al., 2013). Phase change was modeled by Stomakhin et al. (2014) through a dilational/deviatoric splitting of the constitutive model. Yue et al. (2015) adopted Herschel-Bulkley viscoplasticity for foam. Ram et al. (2015) used the Oldroyd-B model for viscoelastic fluids. Fei et al. (2019) developed an analytical plastic flow approach for shear-dependent liquids. Following Drucker-Prager yield criterion, Klár et al. (2016); Daviet and Bertails-Descoubes (2016) modeled sand as continuum granular materials and Tampubolon et al. (2017) further added wetting. Wolper et al. (2019, 2020) captured dynamic fracture using Non-Associated Cam-Clay (NACC) plasticity and damage mechanics. Advocating implicit integrators, stiff plastic materials like metal was simulated with Newton-Krylov MPM (Wang et al., 2020a), while Fang et al. (2019) used Alternating Direction Methods of Multipliers (ADMM) for viscoelasticity and elastoplasticity and Li et al. (2022g) proposed a variational implicit inelasticity formulation.

Inelasticity with Other Discretizations Smoothed Particle Hydrodynamics (SPH) was originally developed for simulating incompressible flow. Clavet et al. (2005) added dynamic-length springs for viscoelasticity. Jones et al. (2014) and Müller et al. (2004) solved Moving

Least Squares (MLS) for elastoplasticity. Gerszewski et al. (2009) first used deformation gradient tensor with multiplicative elastoplastic decomposition in SPH. Alduán and Otaduy (2011) and Yang et al. (2017a) modeled granular materials based on Drucker Prager yielding. Takahashi et al. (2015) used an implicit SPH formulation to simulate viscous fluids. Gissler et al. (2020) developed an implicit SPH snow solver similarly to Stomakhin et al. (2013)'s MPM treatment. Using power diagram-based particle-in-cell (Qu et al., 2022) and MLS-MPM (Hu et al., 2018a), power plastics (Qu et al., 2023) simulated inelastic flow with an XPBD-style Gauss-Seidel solver. Peridynamics (Silling, 2000) defines pairwise forces and integrates particle interactions. He et al. (2017) combined peridynamics with projective dynamics (Bouaziz et al., 2014) and modeled Drucker-Prager plasticity. Chen et al. (2018b) used isotropic linear elasticity with plasticity and simulated fracture.

5.2.3 Method

We start with briefly reviewing XPBD (Macklin et al., 2016), which lets hyperelasticity be governed by Newton's equations of motion through a potential $U(\boldsymbol{x})$: $\boldsymbol{M}\ddot{\boldsymbol{x}} = -\nabla U^T(\boldsymbol{x})$, where $\boldsymbol{M}$ is the mass matrix and $\boldsymbol{x} = [x_1, x_2, ..., x_p]^T$ is the unknown position states. XPBD assumes $U(\boldsymbol{x})$ can be further expressed as $U(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{C}(\boldsymbol{x})^T\boldsymbol{a}^{-1}\boldsymbol{C}(\boldsymbol{x})$, where $\boldsymbol{C} = [C_1(\boldsymbol{x}), C_2(\boldsymbol{x}), ..., C_m(\boldsymbol{x})]^T$ contains m constraints, and $\boldsymbol{a}$ is a diagonal compliance matrix. The elastic internal force $\boldsymbol{f}$ and Lagrange multiplier $\boldsymbol{\lambda}$ are then shown to be

$$\boldsymbol{f} = -\nabla \boldsymbol{C}(\boldsymbol{x})^T\boldsymbol{a}^{-1}\boldsymbol{C}(\boldsymbol{x}), \tag{5.13}$$

$$\boldsymbol{\lambda} = -\tilde{\boldsymbol{a}}^{-1}\boldsymbol{C}(\boldsymbol{x}). \tag{5.14}$$

where $\tilde{\boldsymbol{a}} = \frac{\boldsymbol{a}}{\Delta t^2}$ and Δt is the time step size.

5.2.3.1 Rewriting StVK Elasticity as Constraints

For modeling elasticity, we adopt the St. Venant-Kirchhoff (StVK) model with Hencky strains. As in Klár et al. (2016), Gao et al. (2017), the advantage of this choice is for math/code

simplicity and runtime efficiency – it allows return mapping to have analytical solutions for certain plastic flows, eliminating the need for numerical solutions. The elastoplastic behavior of isotropic materials is characterized in the principal stretch space $\boldsymbol{\Sigma}$ via singular value decomposition (SVD) (Stomakhin et al., 2012) of the deformation gradient $\boldsymbol{F}$. The element's total potential energy Φ can be expressed as $\Phi = V^0\Psi$, where V^0 is an element's rest volume and Ψ is the energy density, assuming piecewise constant element deformations, i.e., one particle has one deformation gradient. For StVK we have energy density Ψ

$$\Psi = \mu \mathrm{tr}\left(\log\left(\boldsymbol{\Sigma}\right)^2\right) + \frac{\lambda}{2}\left(\mathrm{tr}\left(\log\left(\boldsymbol{\Sigma}\right)\right)\right)^2, \tag{5.15}$$

where μ and λ are the Lamé parameters.

To convert Φ into constraints, our first option is to separately handle each term. As done by Macklin and Müller (2021) for Neo-Hookean, by utilizing $\Phi = \frac{1}{2}\frac{1}{\alpha}C^2$, we could define two constraints for the μ and λ term respectively as

$$\alpha_\mu = 1/(2\mu V^0), \text{ and } C_\mu = \sqrt{\mathrm{tr}\left(\log\left(\boldsymbol{\Sigma}\right)^2\right)}; \tag{5.16}$$

$$\alpha_\lambda = 1/(\lambda V^0), \text{ and } C_\lambda = \mathrm{tr}\left(\log\left(\boldsymbol{\Sigma}\right)\right). \tag{5.17}$$

Alternatively, as done by Qu et al. (2023) on power diagrams, we can absorb Lamé parameters from α to C and define a single $\boldsymbol{F}$-dependent constraint through $\Phi = V^0\Psi(\boldsymbol{F}) = \frac{1}{2\alpha}C(\boldsymbol{F})^2$ to achieve

$$\alpha = 1/V^0, \text{ and } C(\boldsymbol{F}) = \sqrt{2\Psi(\boldsymbol{F})}, \tag{5.18}$$

which clearly halves the number of constraints by allowing more nonlinearity in each $C(\boldsymbol{F}(\boldsymbol{x}))$. In practice we find both options effective, and adopt the single-constraint version for efficiency.

5.2.3.2 Gradient is All You Need

Unlike mesh-based representations (Macklin and Müller, 2021) which use linear FEM to model the deformation map $\boldsymbol{x} = \boldsymbol{\phi}(\boldsymbol{X}, t)$ and compute deformation gradient $\boldsymbol{F} = \partial\boldsymbol{\phi}(\boldsymbol{X}, t)/\partial\boldsymbol{X}$ using an undeformed reference state $\boldsymbol{X} \in \Omega^0$, meshless inelastic materials cannot utilize simplex elements due to extreme deformation. We follow Jiang et al. (2016) and Gissler et al. (2020) to derive the time rate of the deformation gradient:

$$\frac{\partial}{\partial t}\boldsymbol{F}(\boldsymbol{X}, t) = \frac{\partial}{\partial t}\frac{\partial\boldsymbol{\phi}}{\partial\boldsymbol{X}}(\boldsymbol{X}, t) = \frac{\partial\boldsymbol{v}}{\partial\boldsymbol{x}}(\boldsymbol{\phi}(\boldsymbol{X}, t), t)\frac{\partial\boldsymbol{\phi}}{\partial\boldsymbol{X}}(\boldsymbol{X}, t), \tag{5.19}$$

with $\boldsymbol{V}(\boldsymbol{X}, t) = \partial\boldsymbol{\phi}(\boldsymbol{X}, t)/\partial t$ being the Lagrangian velocity whose Eulerian counterpart is $\boldsymbol{v}(\boldsymbol{x}, t) = \boldsymbol{V}(\boldsymbol{\phi}^{-1}(\boldsymbol{x}, t), t)$. With time discretization from t^n to t^{n+1} and the assumption that the velocity $\boldsymbol{v}$ at time t^{n+1} being $\boldsymbol{v}^{n+1}(\boldsymbol{x})$ for $\boldsymbol{x} \in \Omega^n$, we have

$$\frac{\partial}{\partial t}\boldsymbol{F}(\boldsymbol{X}, t^{n+1}) = \frac{\partial\boldsymbol{v}^{n+1}}{\partial\boldsymbol{x}}(\boldsymbol{\phi}(\boldsymbol{X}, t^n))\boldsymbol{F}(\boldsymbol{X}, t^n). \tag{5.20}$$

Taking $\frac{\partial}{\partial t}\boldsymbol{F}_p(\boldsymbol{X}_p, t^{n+1}) \approx (\boldsymbol{F}_p^{n+1} - \boldsymbol{F}_p^n)/\Delta t$ for a particle $\boldsymbol{X}_p$ we get

$$\boldsymbol{F}_p^{n+1} = \boldsymbol{F}_p^n + \Delta t\frac{\partial\boldsymbol{v}^{n+1}}{\partial\boldsymbol{x}}(\boldsymbol{x}_p^n)\boldsymbol{F}_p^n = \left(\boldsymbol{I} + \Delta t\frac{\partial\boldsymbol{v}^{n+1}}{\partial\boldsymbol{x}}(\boldsymbol{x}_p^n)\right)\boldsymbol{F}_p^n \tag{5.21}$$

as the evolution of $\boldsymbol{F}_p^{n+1}$ given $\boldsymbol{v}^{n+1}$ and $\boldsymbol{F}_p^n$. With updated Lagrangian (De Vaucorbeil et al., 2020), the reference space is thus always Ω^n and there is no need to store Ω^0; see Figure 5.16. Therefore, to express constraints as functions of positions $C(\boldsymbol{F}(\boldsymbol{v}^{n+1}(\boldsymbol{x}^{n+1})))$, *robustly estimating $\partial\boldsymbol{v}/\partial\boldsymbol{x}$ ($\forall x \in \Omega^n$) and its derivative is all one needs.*

Accurately estimating meshless velocity gradients is generally challenging. Most meshless shape functions require a dense neighborhood to fulfill the kernel's normalization condition. Known as neighborhood deficiency, significant accuracy degradation would occur especially for first-order derivatives (such as velocity gradient) in sparse regions. Kernel gradient correction (Bonet and Lok, 1999) and the reproducing kernel particle method (Liu et al., 1995) are examples of strategies for mitigating this problem. Here, we adopt Wendland

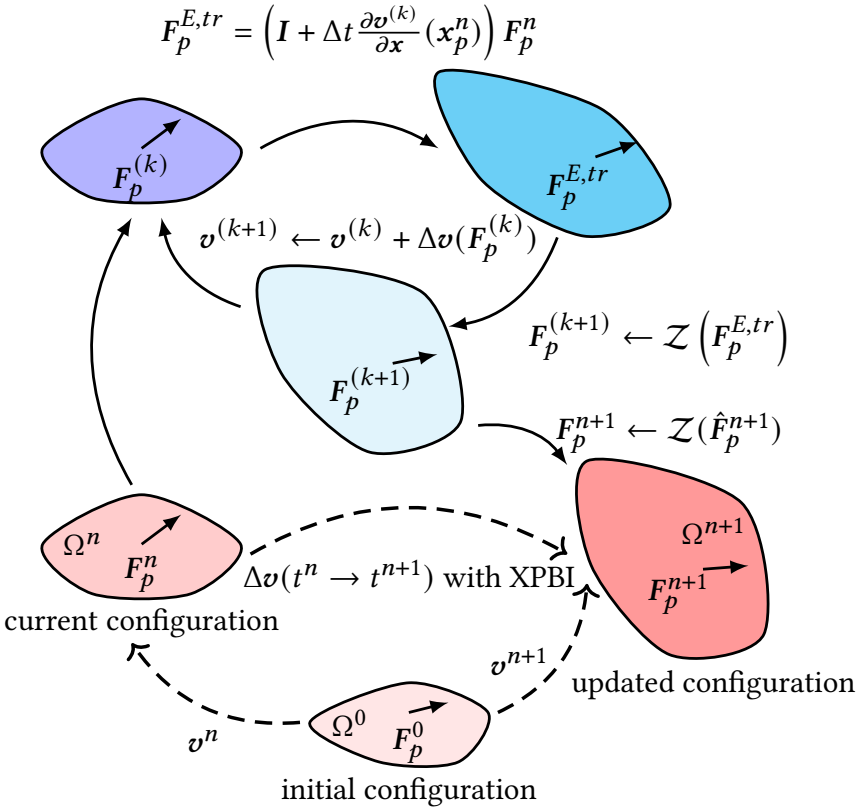


Figure 5.16: **Deformation Gradient Evolution.** The dotted line (**bottom**) illustrates the evolution of the deformation gradient in the updated Lagrangian view, transitioning from $\boldsymbol{F}_p^0$ (initial configuration) to $\boldsymbol{F}_p^{n+1}$ (updated configuration), with $\boldsymbol{F}_p^n$ (current configuration) serving as the reference state. This facilitates tracking large deformations. The solid line loop (**top**) depicts iterations of our XPBI algorithm to simulate $t^n \rightarrow t^{n+1}$, alternating between an XPBD iteration and a fixed point iteration. During iteration k, $\boldsymbol{F}_p^{E,tr}$ is first estimated based on the current gradient of $\boldsymbol{v}^{(k)}$ (Section 5.2.3.2). Then, plasticity is applied through projection to obtain $\boldsymbol{F}_p^{(k+1)}$ (Section 5.2.3.3), and finally, $\boldsymbol{v}^{(k+1)}$ is updated by solving constraints (Section 5.2.4.1).

kernels (Wendland, 1995) for the standard SPH kernel W and ∇W and the reweighting-based kernel gradient correction (Bonet and Lok, 1999). The correction matrix $\boldsymbol{L}_p$ is defined as

$$\boldsymbol{L}_p = \left(\sum_b V_b^n \nabla W_b(\boldsymbol{x}_p) \otimes (\boldsymbol{x}_b - \boldsymbol{x}_p) \right)^{-1}, \tag{5.22}$$

with SVD based pseudo inverse $\boldsymbol{A}^{-1} = \boldsymbol{V}\boldsymbol{\Sigma}^{-1}\boldsymbol{U}^T$ to avoid singularities when calculating ill-conditioned matrix inverses for numerical stability, where $V_b^n = V_b^0 \det(\boldsymbol{F}_b^n)$ is the time n volume of the neighborhood particle b. As in many SPH methods, the correction is

indispensable for maintaining simulation stability. See Westhofen et al. (2023) for discussions of similarly viable gradient estimation choices. Our discrete velocity gradient at particle p in Ω^n is then

$$\frac{\partial \boldsymbol{v}}{\partial \boldsymbol{x}}(\boldsymbol{x}_p^n) = \sum_{b \neq p} V_b^n (\boldsymbol{v}_b - \boldsymbol{v}_p) \left(\boldsymbol{L}_p \nabla W_b(\boldsymbol{x}_p^n) \right)^T, \tag{5.23}$$

which is also adopted in Gissler et al. (2020). Combining the gradient estimation with Equation (5.21) and differentiating per-particle constraint $C_p(\boldsymbol{F}_p)$ (Equation (5.18)) reveals

$$\nabla_{\boldsymbol{x}_b} C_p|_{b \neq p} = V_b^n \frac{\partial C_p}{\partial \boldsymbol{F}_p} \boldsymbol{F}_p^{nT} (\boldsymbol{L}_p \nabla W_b(\boldsymbol{x}_p^n)), \tag{5.24}$$

$$\nabla_{\boldsymbol{x}_p} C_p = -\sum_{b \neq p} \nabla_{\boldsymbol{x}_b} C_p, \tag{5.25}$$

which provides us all necessary constraint derivatives in XPBD.

5.2.3.3 Implicit Plasticity

Plasticity in continuum mechanics is typically solved with return mapping, denoted as $\mathcal{Z}(\cdot)$, which adjusts strains according to a plastic flow rule. It projects an elastic predictor $\boldsymbol{F}^{E,tr}$ onto the yield surface to ensure an inequality constraint on the stress.

To make plasticity implicit, we propose to alternate between (1) an XPBD iteration with a projected stress and (2) a stress projection. This is essentially a fixed point iteration similarly to Li et al. (2022g):

$$\boldsymbol{F}_p^{(k+1)} \leftarrow \mathcal{Z} \left(\boldsymbol{F}_p^{E,tr} \left(\boldsymbol{v}^{(k)} \left(\boldsymbol{F}_p^{(k)} \right) \right) \right), \tag{5.26}$$

where $\boldsymbol{F}_p^{E,tr}$ is the trial elastic deformation gradient and $\boldsymbol{v}^{(k)}(\boldsymbol{F}_p^{(k)})$ is the updated velocity by previous k XPBD iterations based on $\boldsymbol{F}_p^{(k)}$ in the previous iteration. In contrast to Li et al. (2022g)'s fixed point iteration on $\mathcal{Z}$ which functions as an independent outer loop of a full Newton optimization, our design establishes a fixed point on $\boldsymbol{F}$, updating variables

Algorithm 6 Simulating $t^n \rightarrow t^{n+1}$ with XPBI

Neighbor search using $\boldsymbol{x}^n$ ▷ Section 5.2.4.2
Evaluate kernel gradient correction $\boldsymbol{L}_p$ ▷ Section 5.2.3.2
$\boldsymbol{v} \leftarrow \boldsymbol{v}^n + \Delta t \boldsymbol{M}^{-1} \boldsymbol{f}_{\text{ext}}$
$\boldsymbol{\lambda} = \boldsymbol{0}$
for number of XPBD iterations
  for all $p \in \{C_p\}$ looping with colored Gauss-Seidel ▷ Section 5.2.4.3
    $(\nabla \boldsymbol{v})_p \leftarrow \text{evaluateVelocityGradient}(\boldsymbol{x}_p^n, \boldsymbol{v}_p)$ ▷ Section 5.2.3.2
    $\boldsymbol{F}_p \leftarrow (\boldsymbol{I} + \Delta t (\nabla \boldsymbol{v})_p) \boldsymbol{F}_p^n$
    $\boldsymbol{F}_p \leftarrow \mathcal{Z}(\boldsymbol{F}_p)$ ▷ Section 5.2.3.3
    if $C_p(\boldsymbol{F}_p) \neq 0$ **then**
      $\Delta \lambda_p = \frac{-C_p - \tilde{\alpha}\lambda}{\sum_{i=1}^{N} \frac{1}{m_i} |\nabla_{\boldsymbol{x}_i} C_p(\boldsymbol{x})|^2 + \tilde{\alpha}}$ ▷ Section 5.2.3.1
      $\lambda_p \leftarrow \lambda_p + \Delta \lambda_p$
      $\Delta \boldsymbol{v} = \frac{1}{\Delta t} \boldsymbol{M}^{-1} \nabla C_p(\boldsymbol{x})^T \Delta \lambda_p$
      $\boldsymbol{v} \leftarrow \boldsymbol{v} + \Delta \boldsymbol{v}$
    end if
  end
  for all $i \in \{C_i\}$ looping with colored Gauss-Seidel
    $\Delta \boldsymbol{v} = \frac{1}{\Delta t} \boldsymbol{M}^{-1} \nabla C_i(\boldsymbol{x}^n + \Delta t \boldsymbol{v})^T \Delta \lambda_i$ (e.g., collision) ▷ Section 5.2.4.4
    $\boldsymbol{v} \leftarrow \boldsymbol{v} + \Delta \boldsymbol{v}$
  end
end
$\boldsymbol{v}^{n+1} \leftarrow \boldsymbol{v}$
Perform XSPH smoothing of $\boldsymbol{v}^{n+1}(\boldsymbol{x}^n)$ ▷ Section 5.2.4.4
Update $\boldsymbol{F}^{n+1}$ and apply constitutive models ▷ Section 5.2.4.5
$\boldsymbol{x}^{n+1} \leftarrow \boldsymbol{x}^n + \Delta t \boldsymbol{v}^{n+1}$
Note: $\alpha = 1/V_p^0$ and $\tilde{\alpha} = \alpha / \Delta t^2$ for each constraint.

directly impacted by the fixed point iteration *within* XPBD iterations; see Figure 5.16 top and Algorithm 6. Resultingly, our implicit plasticity treatment introduces negligible extra cost on top of what implicit elasticity already necessitated.

Nonetheless, convergence in fixed-point iterations depends on an initial guess sufficiently close to the solution, among other conditions of the implicit function. In this paper, we do not monitor quantitative plasticity convergence since few XPBD iterations are needed for visually plausible results. We do emphasize the importance of implicit plasticity and compare it with a semi-implicit treatment which only applies plasticity at the end of a time step; see Section 5.2.5.1.

5.2.4 Algorithm

Here we detail the XPBI pipeline and its seamlessly integration into existing XPBD. Our pseudocode for advancing a time step using velocity-based XPBD is summarized in Algorithm 6.

5.2.4.1 Algorithm Overview

Similarly to MPM, we use material particles to discretize the continuum. Each particle p is governed by a constitutive model-induced constraint C_p (Equation (5.18)) and a plastic return mapping operator $\mathcal{Z}$. We use C_p to denote particle-wise inelasticity constraints ($|\{p\}| = N = \#$ particles) and C_i to denote traditional PBD constraints ($|\{i\}| = M = \#$ number of all other constraints).

Due to our dependency on velocity gradients, it is more natural to reparametrize XPBD with velocities rather than positions as primary unknown variables. Closely resembling position-based XPBD, we solve for velocities $\boldsymbol{v}^{n+1}$ and Lagrange multipliers $\boldsymbol{\lambda}^{n+1}$ that satisfies

$$\boldsymbol{M}(\boldsymbol{v}^{n+1} - \tilde{\boldsymbol{v}}^n) - \nabla \boldsymbol{C}(\boldsymbol{v}^{n+1})^T \boldsymbol{\lambda}^{n+1} = \boldsymbol{0}, \tag{5.27}$$

$$\boldsymbol{C}(\boldsymbol{v}^{n+1}) + \tilde{\boldsymbol{\alpha}} \boldsymbol{\lambda}^{n+1} = \boldsymbol{0}, \tag{5.28}$$

by updating per-particle inelastic constraint C_p's corresponding Lagrangian multiplier λ_p with

$$\Delta \lambda_p = \frac{-C_p - \tilde{\alpha}_p \lambda_p}{\sum_{b=1}^{N} \frac{1}{m_b} |\nabla_{\boldsymbol{x}_b} C_p(\boldsymbol{x})|^2 + \tilde{\alpha_p}}, \tag{5.29}$$

where $\alpha_p = 1/V_p^0$ and $\tilde{\alpha}_p = \alpha_p / \Delta t^2$. The velocity update is given by:

$$\Delta \boldsymbol{v} = \left(\boldsymbol{M}^{-1} \nabla \boldsymbol{C}(\boldsymbol{x})^T \Delta \boldsymbol{\lambda} \right) / \Delta t. \tag{5.30}$$

The velocities and multipliers are jointly updated by a colored Gauss-Seidel iteration (see

Section 5.2.4.3 and Algorithm 6):

$$\lambda_p^{(k+1)} \leftarrow \lambda_p^{(k)} + \Delta\lambda_p, \quad \boldsymbol{v}^{(k+1)} \leftarrow \boldsymbol{v}^{(k)} + \Delta\boldsymbol{v}. \tag{5.31}$$

Note that by selecting velocities as our primary unknown variables, our deformation gradient update (as derived in Equation (5.20) and Equation (5.21)) within the Gauss-Seidel iteration allows us to directly evaluate the velocity gradient and particle-wise constraints in Ω^n using interpolation kernels defined at t^n, aligning with a typical MPM approach (Jiang et al., 2016). Conversely, using positions as primary variables could introduce uncertainties regarding whether to update the kernels during implicit iterations, which is an intriguing area for future exploration. For collisions with standard PBD materials, as illustrated in the loop over C_i in Algorithm 6, we follow the standard PBD approach, where constraints are directly evaluated using collision kernels defined by the latest candidate positions during the iterations, ensuring accurate prediction of potential collisions.

5.2.4.2 Particle Neighbor Search

We reconstruct the neighbor information for each particle at the beginning of each timestep, similarly to Macklin and Müller (2013). A comprehensive overview of CPU- and GPU-based neighborhood search methods is surveyed by Ihmsen et al. (2014). Each material particle is assigned the same kernel radius in our discretization scheme. We adopt a uniform spatial-grid-based method for neighborhood searches following Hoetzlein (2014). Particles are spatially stored in cells while neighbor lists $\mathcal{N}_p = \{b \mid \|x_p - x_b\|_2 \leq k\}$ are determined by querying adjacent grid cells using the Wendland kernel's support radius $k = 2r$, and r is the SPH particle kernel radius. In addition to the total number of particles $N = |\{p\}|$, the total number of particle neighbors $\sum |\mathcal{N}_p|$ is also critical for performance, as it determines the complexity of calculating inelastic constraints. We summarize the statistics for $\sum |\mathcal{N}_p|$ in Table 5.3.

5.2.4.3 Colored Gauss-Seidel

Original PBD and XPBD frameworks solve constraints iteratively using nonlinear Gauss-Seidel (Müller et al., 2007; Macklin et al., 2014, 2016). In contrast, Macklin and Müller (2013) adopted a Jacobi-style iteration for fluids, solving each constraint independently to enhance parallelism. We found that for high resolution and often high stiffness simulations considered in this paper, Jacobi iterations too slowly propagate information and often suffer from non-convergence (also noted by Macklin et al. (2014)). Thus we implement colored Gauss-Seidel to maximize convergence, parallelism, and GPU throughput. We assign particles into cells with $\Delta x = 2r$ (Section 5.2.4.2). 2^d colors are specified for eliminating dependencies between constraints in a d-dimensional simulation. We process all cells of the same color in parallel while constraints $C_p \in c_i$ corresponding to all particles in the same cell c_i are computed serially.

The efficiency of this implementation heavily depends on the average Particle Per Cell (PPC), as particles within the same cell are traversed sequentially. To optimize PPC and ensure an even particle distribution, we utilize Poisson disk sampling (Bridson, 2007) during the initial particle placement.

5.2.4.4 XSPH and Position Correction

The goal of eXtended Smoothed Particle Hydrodynamics (XSPH) is to incorporate artificial viscosity for mitigating nonphysical oscillations in dynamics observed in SPH-based simulations, which is more observable when the material is stiff and the constraints become hard to solve. We follow Schechter and Bridson (2012)'s simpler XSPH-style noise damping after velocity update by blending in surrounding particle velocities in Ω^n:

$$\boldsymbol{v}_p^{n+1} \leftarrow \boldsymbol{v}_p^{n+1} + c \sum_b V_b^n (\boldsymbol{v}_b^{n+1} - \boldsymbol{v}_p^{n+1}) W_b(\boldsymbol{x}_p^n). \tag{5.32}$$

XSPH encourages smooth and coherent motion for inelastic materials in this paper. We use dimensionless $c = 0.01$ in all examples.

In particle-based simulations, including but not limited to FLIP (Brackbill and Ruppel, 1986; Zhu and Bridson, 2005) and APIC (Jiang et al., 2015a), non-physical particle clumping and uneven particle distributions are common issues due to accumulation of advection errors. While MPM is insensitive to the problem due to its structural grid nature, it is crucial for pure particle-based methods like SPH and our approach to maintain reasonably even particle distributions. Without doing so can strongly impair simulation quality and convergence, particularly when material stiffness is high.

Various techniques in graphics have been proposed to address this by shifting particle positions (Ando et al., 2012; Ando and Tsuruno, 2011; Kugelstadt et al., 2019). However, we point out that while these methods work well for fluids, they are problematic for updated Lagrangian simulations because such positional shifting is transparent to the evolution of deformation gradients, leading to discrepancy between the positions of points and their perceived deformations. Fortunately within XPBD we can directly adopt a point-point distance constraint (Macklin et al., 2014)

$$C(\boldsymbol{x}_p, \boldsymbol{x}_b) = \|\boldsymbol{x}_p - \boldsymbol{x}_b\|_2 - r + \epsilon \geq 0 \tag{5.33}$$

inside nonlinear iterations for all the neighborhood particle pairs. Here r represents the particle kernel radius and ϵ is a small gap threshold used to determine when corrective action is needed for nearby particle pairs. We set ϵ to $0.25r$.

5.2.4.5 Deformation Gradient Update

Strain variables within XPBD iterations are temporary. After arriving at the post-XSPH velocity $\boldsymbol{v}^{n+1}$, we update both the deformation gradient state and position state using the same velocity – an essential subtlety to maintain their consistency:

$$\boldsymbol{F}_p^{n+1} = \mathcal{Z}\left(\left(\boldsymbol{I} + \Delta t \frac{\partial \boldsymbol{v}^{n+1}}{\partial \boldsymbol{x}}(\boldsymbol{x}_p^n)\right)\boldsymbol{F}_p^n\right), \tag{5.34}$$

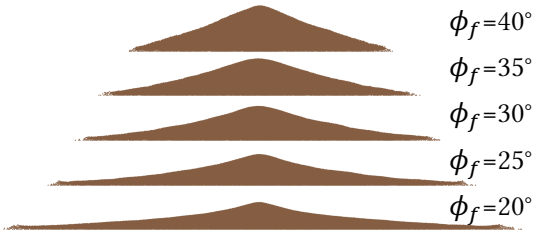


Figure 5.17: We simulate sand collapsing with varying friction angles ϕ_f.

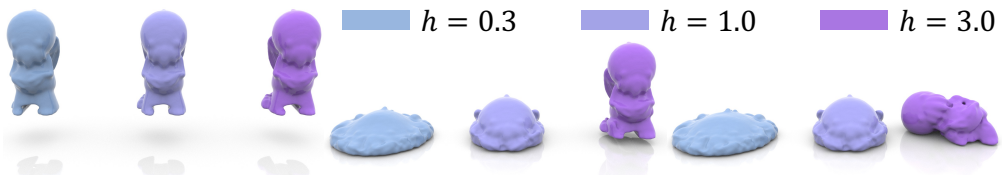


Figure 5.18: XPBI simulates Hershel-Bulkley shear thinning ($h = 0.3$), viscoplastic ($h = 1.0$), and shear thickening materials ($h = 3.0$), where h controls a power law flow rate detailed in (Yue et al., 2015).

where inelastic return mapping is also applied to ensure the stored elastic deformation gradient is within the yield region.

5.2.5 Results

Here we evaluate and benchmark our eXtended Position-based Inelasticity (XPBI) framework in terms of visual results against traditional XPBD and MPM methods, as detailed in Section 5.2.5.1. Additionally, we present various demonstrations in Section 5.2.5.2 that illustrate XPBI's effective handling of diverse phenomena. We use Intel Core i9-14900KF CPU with 32GB memory and NVIDIA GeForce RTX 4090. We model common inelastic materials including Cam-Clay (NACC) (Wolper et al., 2019) snow and fracture, Drucker-Prager (Klár et al., 2016; Tampubolon et al., 2017) sand, Von Mises (Li et al., 2022g) plasticine and metal, and Herschel-Bulkley (Yue et al., 2015) foam.

5.2.5.1 Evaluation

Intuitive Parameters We simulate sand collapsing with varying friction angles ϕ_f in Figure 5.17. Our method reproduces characteristic piling shapes. In Figure 5.18, we compare

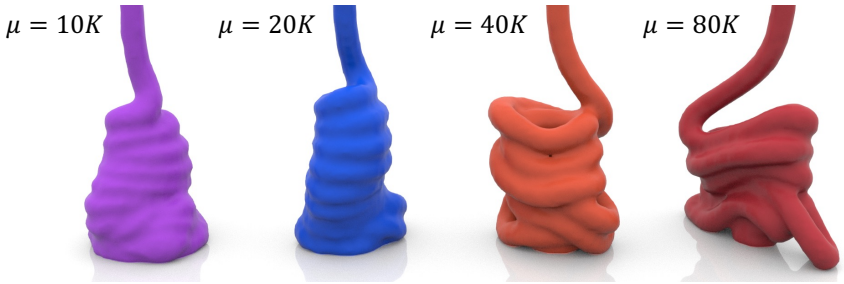


Figure 5.19: XPBI effectively captures viscoplastic coiling; as the shear modulus μ increases from left to right, coils become more elastic.

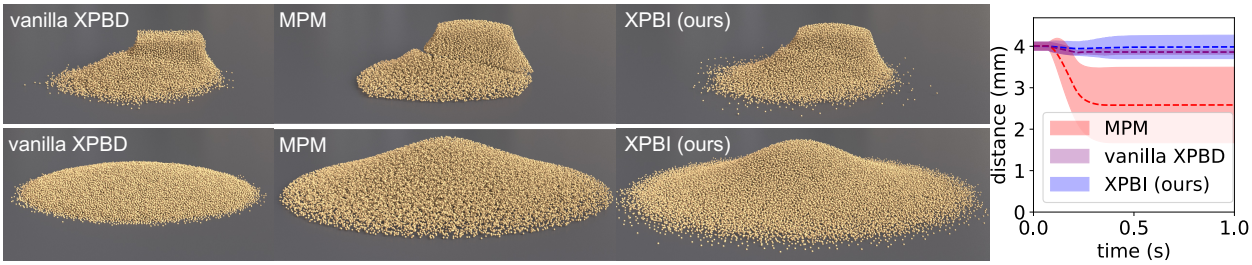


Figure 5.20: **Vanilla XPBD v.s. MPM v.s. XPBI.** We simulate *two sand blocks collide* (**top**) and *sand column collapse* (**bottom**) with vanilla XPBD (Macklin et al., 2014, 2016) (**left**), MPM (Klár et al., 2016) (**middle**), and XPBI (**right**).

viscoplastic, shear thinning, and shear thickening materials by only altering the Herschel-Bulkley power parameter h. Upon impact with a ground plane, the shear thickening material exhibits low flow rates under high stress, behaving elastically and bouncing off. Conversely, the shear thinning material flows immediately due to its higher flow rate. Similarly we can easily control the fluidity of viscoplastic goo (Figure 5.19). A smaller μ gives a more fluid-like appearance, while a larger μ leads to more elastic behavior.

Comparisons to Vanilla XPBD and MPM We compare XPBI sand with both vanilla XPBD (Macklin et al., 2014, 2016), which employs a point-wise friction model, and explicit MPM with Drucker-Prager plasticity (Klár et al., 2016); see Figure 5.20. This comparison includes simulations of *two sand blocks collide* (top) and *sand column collapse* (bottom). All methods apply a timestep of $\Delta t = 0.1$ ms, with identical initial sampling positions for all particles across the methods. The vanilla XPBD (left) approach fails to accurately replicate the correct friction angle upon sand settling. While both MPM (middle) and XPBI (right) successfully model the continuum behavior of sand, our method achieves a more uniform

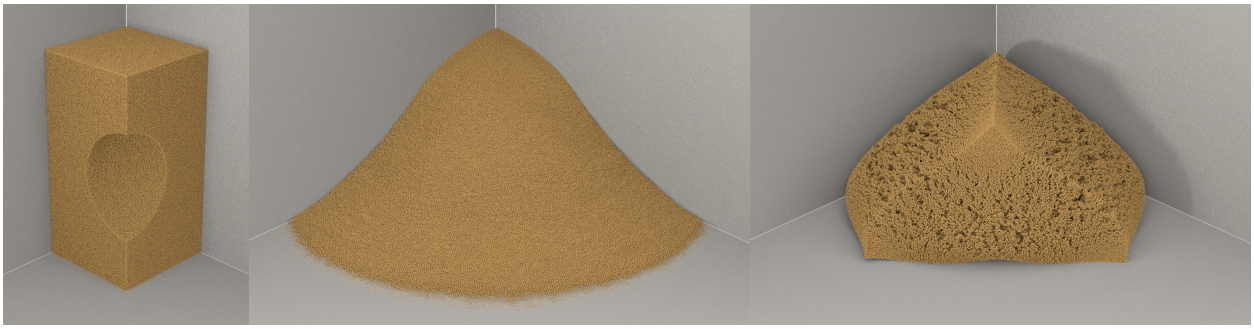


Figure 5.21: Comparison on notched sand block fall. Inital (**left**), our fully implicit treatment (**middle**), and semi-implicit plasticity (**right**).

particle distribution, avoiding the sparsity, clumping, and artificial grid Δx-gap phenomena typically caused by MPM solvers. For a quantitative analysis, we also plotted the average distance between each particle and its nearest neighbor per frame, displayed on the far right of Figure 5.20. Each particle was initially positioned at intervals equal to the particle radius. The average distance relative to the initial state in the MPM decreases rapidly post-collision, accompanied by an increase in the variance of the distance, whereas our method maintains both the relative distance and variance stably throughout the simulation, indicating a more consistent particle distribution. It is also noteworthy that our method aligns more closely with the approach of Yue et al. (2018) compared to MPM, opting for Discrete Element Method (DEM) to capture more discrete behaviors near the free surface.

Implicit Plasticity We evaluate our fully implicit plasticity treatment by replacing it with a semi-implicit method, which only performs return mappings at the end of the time steps as in Stomakhin et al. (2013) Section 5.2.4.5), while XPBD iterations only address elasticity. As shown in Figure 5.21, the semi-implicit approach (right) can lead to severe artifacts. This occurs because the forces generated by stresses outside the yield surface cause the continuum to behave more like a purely elastic body. This artifact results from the semi-implicit method's failure to account for plasticity during the XPBD solve, leading to an overestimation of the material's resistance to tensile deformation. In contrast, our method (middle) fully incorporates plasticity in the XPBD iterations and avoids such artifacts.

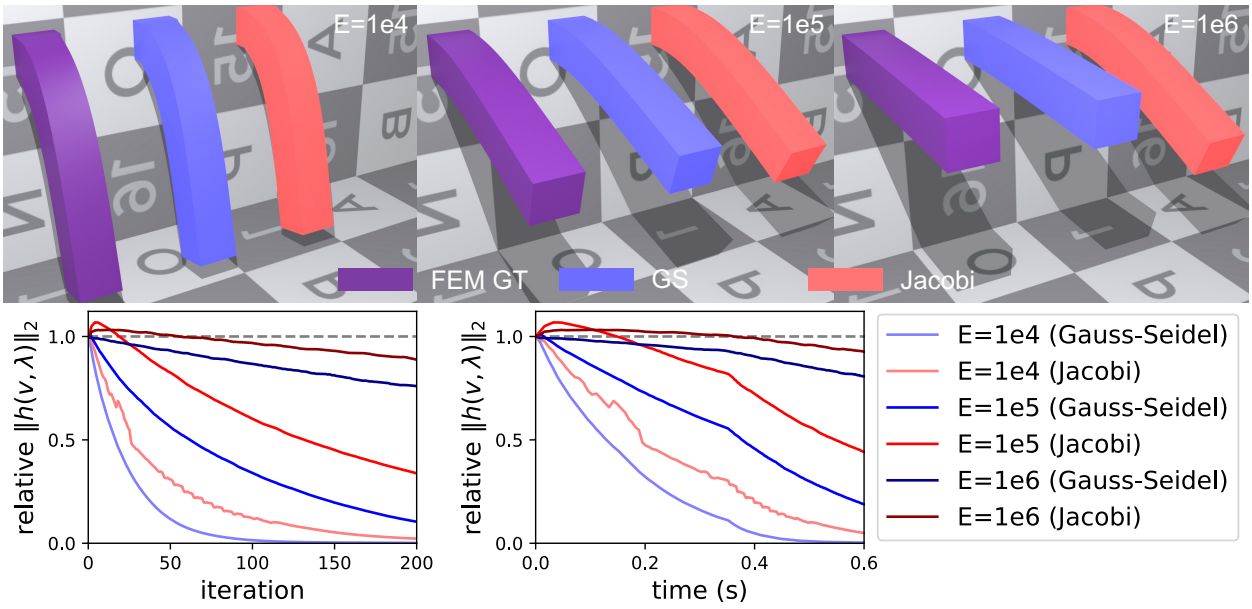


Figure 5.22: Cantilever beams modeled with StVK constitutive model using both our method and FEM ground truth with varying stiffness (**top**). Relative residual errors $\boldsymbol{h(v,\lambda)} = \boldsymbol{C(v)} + \tilde{\boldsymbol{a}}\boldsymbol{\lambda}$ with respect to iteration (**bottom left**) and runtime (**bottom right**) for a single frame from the above examples.

Convergence We study the convergence of our method using cantilever beams of varying stiffness, $E = 10^4\,\text{Pa}, 10^5\,\text{Pa}, 10^6\,\text{Pa}$, respectively (Figure 5.22). We set the density at $100\,\text{kg/m}^3$ and the timestep at $\Delta t = 5\,\text{ms}$. We also compare the relative residual errors of the Gauss-Seidel and Jacobi solvers in our method, as well as an implicit FEM ground truth, with respect to both iteration and time. XPBI with Gauss-Seidel can converge stably with a large timestep. In contrast, the Jacobi solver only converges with softer materials and struggles with high stiffness. This is consistent with observations about XPBD in prior work. Given that the materials discussed in our paper are predominantly very stiff, we opted for grid-colored Gauss-Seidel as our solver. Although using a large timestep is feasible with sufficient iterations, it becomes cost-inefficient if too many iterations are required. Thus, following Macklin et al. (2019), we employ a small timestep.

Position Correction To validate the importance of position correction, we conducted a hydraulic test on a highly stiff aluminum wheel, as shown in Figure 5.23. Without position correction, areas with significant deformation and stress suffer from gradient estimation errors

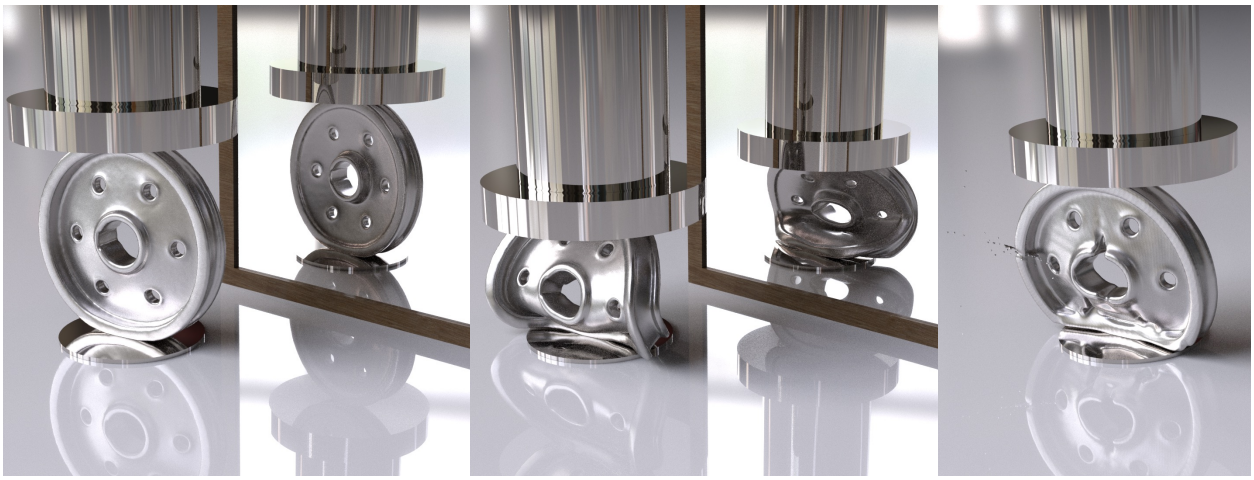


Figure 5.23: Hydraulic test on a stiff aluminum wheel: initial (**left**), w/ (**middle**) and w/o (**right**) position correction. The simulation suffers from artificial fracture and instability without our correction.

due to uneven particle distribution, resulting in artificial fractures and eventually simulation instability. With position correction, however, we can reliably simulate high-stiffness materials under extensive deformation. This example also shows our capability in animating metal ductility using Von Mises plasticity.

In addition to the distance constraint, other position correction strategies can also be employed. For instance, Takahashi and Lin (2019a) demonstrated that the density constraint is effective in addressing particle clustering while preserving volume. In Figure 5.24, we quantitatively compare the distance and density constraints using the same setup as in Figure 5.20. Although both constraints are effective in maintaining maximum density during simulation, the distance constraint better resolves the distance between neighboring particles, improving both mean and variance, crucial for integration stability. We prefer the distance constraint for its simplicity.

Scalability To demonstrate the scalability of our method, we simulate viscoplastic monsters hitting the ground using 8K, 56K, 400K, and 3M particles, respectively. We maintain a constant simulation time step of $\Delta t = 0.1$ ms, perform 10 iterations of XPBD per substep, and apply consistent material parameters across all simulations. Our method consistently replicates material behavior at varying resolutions, as depicted in Figure 5.25. We measure and

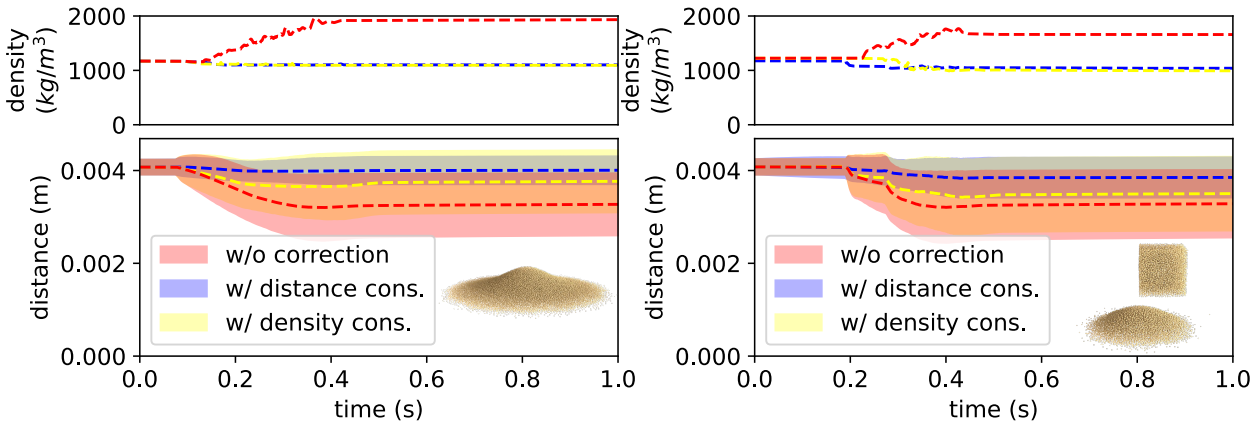


Figure 5.24: Comparison of distance and density constraint. We plot the maximum density (**top**), the average distance to the nearest neighbor and respective standard deviations (**bottom**) for the settings described in Figure 5.20.

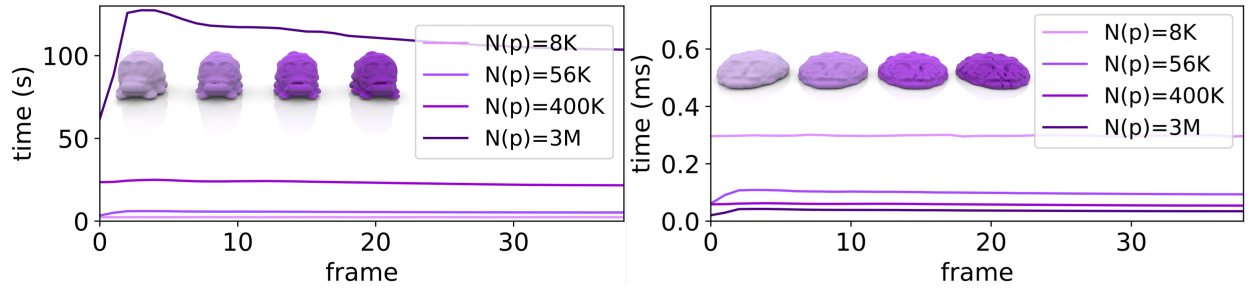


Figure 5.25: We simulate viscoplastic monsters falling to the ground using varying numbers of particles. We plot the computation time for each frame for 8K, 56K, 400K, and 3M (**left**) particles, demonstrating consistent behaviors across different particle counts. We also plot the average cost per particle (**right**). The running overhead of our algorithm decreases significantly as the number of particles increases, showing strong superlinear scalability.

plot the computation time for each individual frame (left) alongside the average computation time per particle (right). The average computation times per particle for 8K, 56K, 400K, and 3M are 0.30 ms, 0.098 ms, 0.058 ms, and 0.037 ms, respectively. These results highlight strong scalability; as the number of particles increases, the colored Gauss-Seidel solver can more effectively exploit GPU resources, significantly reducing the overall computational overhead per particle.

Timing Breakdown Figure 5.26 illustrates the GPU computational cost breakdown for the *Hourglass* simulation example. In the breakdown, *Collision Detection* refers to the time spent on constructing the LBVH (Karras, 2012) and querying for collision between

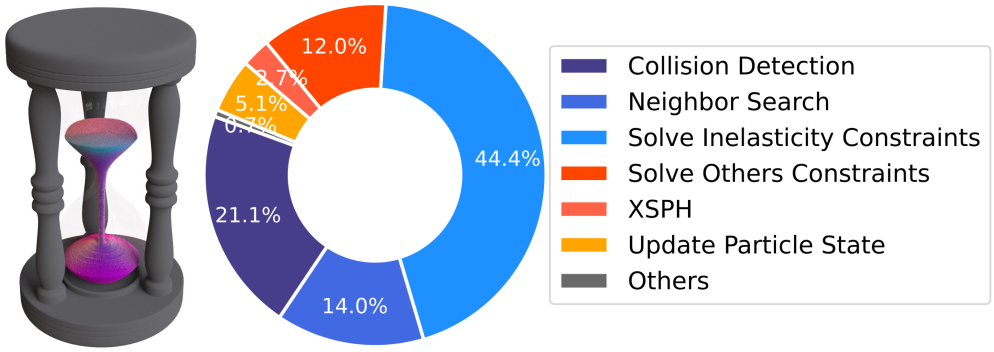


Figure 5.26: A typical breakdown of the total computational cost of our framework. We take the *Hourglass* example (Figure 5.33) for demonstration.

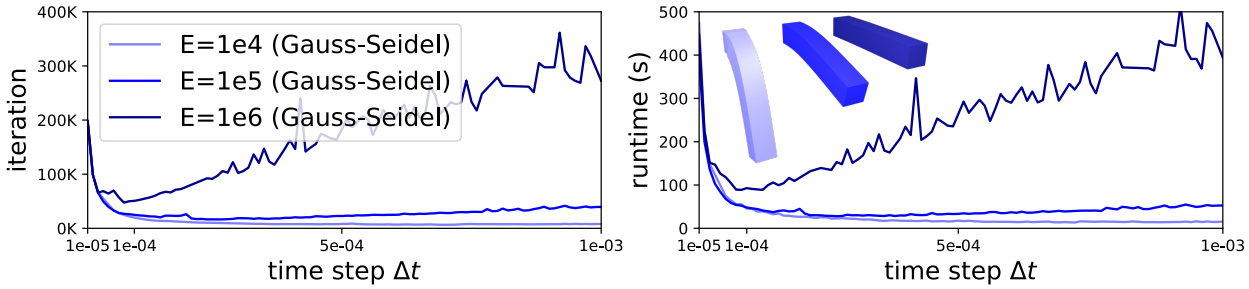


Figure 5.27: Ablation on varying time step sizes. Total XPBD iterations (**left**) and runtime (**right**) required for all steps to converge to a fixed residual error for 1s simulation time with varying time step sizes.

point-triangle pairs. *Neighbor Search* covers the time taken to build the background grid and neighborhood list by querying adjacent cells (see Section 5.2.4.2 for details). *Solve Inelasticity Constraints* involves our colored Gauss-Seidel solver for inelasticity constraints, including fixed-point implicit plasticity treatment. *Solve Other Constraints* accounts for the time spent on resolving all other constraints, such as point-triangle distance constraints for boundary conditions, position correction, as well as stretching, bending, and density constraints in other examples. *XSPH* and *Update Particle State* are detailed in Section 5.2.4.4 and Section 5.2.4.5, respectively. The majority of our framework's computational time is spent on solving inelasticity constraints, while additional stability enhancements like XSPH and position correction contribute a relatively small overhead.

Choice of Time Step While our method with colored GS solver can converge with a time step 5× larger, as noted by Macklin et al. (2019), smaller time steps are generally

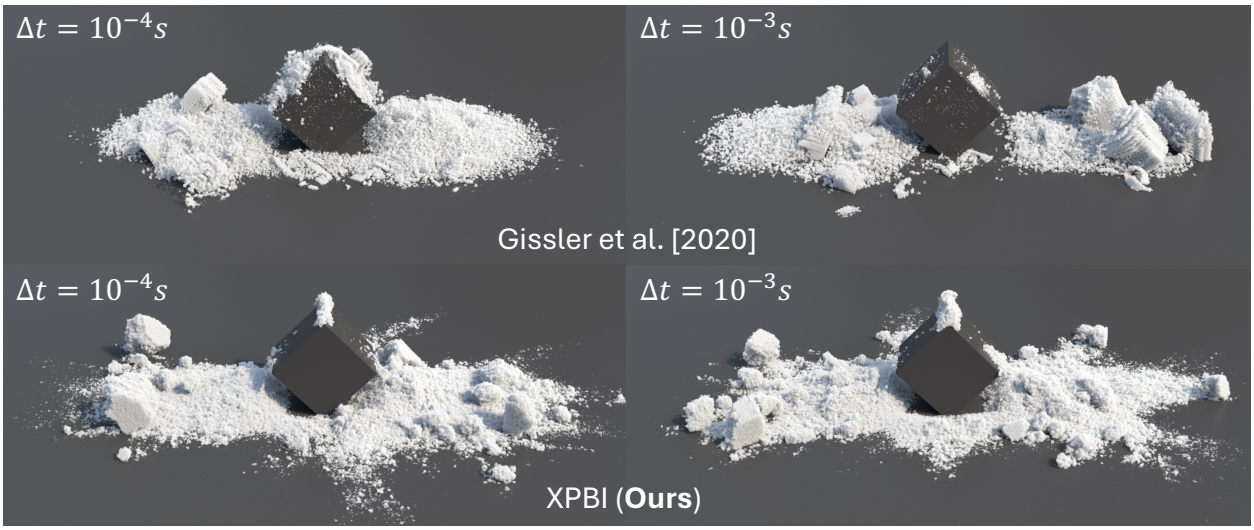


Figure 5.28: Comparison with Gissler et al. (2020). The semi-implicit plasticity approach in Gissler et al. (2020) (**top**) exhibits timestep-dependent behavior, whereas our method (**bottom**) demonstrates consistent behavior across different timestep sizes.

preferable due to the nonlinear increase in GS iterations needed for convergence. However, we emphasize that reducing to just one iteration per time step is sub-optimal in our scenario, as shown in Figure 5.26, where collision detection, neighbor search, XSPH, and particle state updates occur once per time step, accounting for about 1/3 of the total computation time. A timestep that is too small increases the overhead of these operations, with minimal benefit to GS convergence. We study the efficiency of different time step sizes by applying the same setting as in Figure 5.22, this time fixing the residual error threshold ϵ_E and ensuring that each timestep converges under the given threshold, $\|h(v, \lambda)\|_2 \leq \epsilon_E$, with adaptive XPBD iterations. The ϵ_E values are chosen based on the residual errors observed when the cantilever beams exhibit visually identical behaviors to the FEM ground truth for each stiffness E, respectively. We measure the total XPBD iterations and runtime required relative to the timestep size for a 1-second simulation. As shown in Figure 5.27, when stiffness is high, the total number of iterations required for convergence increases with the timestep size. Interestingly, when the timestep is sufficiently small, the total XPBD iterations required actually increases, as each timestep necessitates at least one iteration, and the runtime's growth rate rises further due to the additional overhead per timestep. In practice, we select a Δt between 5×10^{-5} s and 2×10^{-4} s.

Table 5.3: **Parameters and Statistics.** We summarize the parameters and timing statistics, including maximum particle numbers, the Wendland kernel radius, the average total number of particle neighbors per substep, the average time per frame, the XPBD iterations per substep, and the time step size Δt for various demos described in Section 5.2.5.2. Material-related parameters are detailed in the last two columns. In addition to the basic settings of the material (density ρ, Youngs Modulus E, and Poisson Ratio ν), we include model-specific parameters arranged as follows: 1) NACC: $(\rho, E, \nu, \alpha_0, \beta, \xi, M)$; 2) DP: $(\rho, E, \nu, \phi_f, c_0)$; 3) VM: $(\rho, E, \nu, \sigma_\gamma)$; and 4) HB: $(\rho, E, \nu, \sigma_\gamma, h, \eta)$. See references for detailed explanations of these parameters.

demo	particle #	radius	ave $\sum \lvert\mathcal{N}_p\rvert$	ave sec/frame	iter #	Δt_{frame}	Δt_{step}	material	material parameters
(Figure 5.29) Noodles	1.18M	1/256	28.7M	46.3	10	1/40	1×10^{-4}	VM	$(1, 2 \times 10^4, 0.3, 76.9)$
(Figure 5.31) Cloth	1.10M	1/512	19.9M	24.3	10	1/100	5×10^{-5}	HB	$(100, 14754, 0.475, 50, 1, 10)$
(Figure 5.32) Camponotus	1.12M	1/1024	32.9M	37.3	10	1/100	4×10^{-5}	NACC	$(2, 2 \times 10^4, 0.35, -0.02, 0.5, 1, 2.36)$
(Figure 3.15) Dam Breach	4.00M	1/384	156.2M	138.8	7	1/24	2.5×10^{-4}	DP	$(1, 400, 0.4, 30, 0.0007)$
(Figure 5.33) Hourglass	1.01M	1/1024	17.0M	30.9	5	1/24	1×10^{-4}	DP	$(1, 3.537 \times 10^5, 0.3, 35, 0)$
(Figure 4.30) Hitman	1.05M	1/512	32.5M	38.9	10	1/100	4×10^{-5}	NACC	$(4, 2 \times 10^4, 0.3, -0.005, 0.05, 30, 1.85)$
(Figure 4.30) Snow Dive	2.48M	1/512	64.1M	78.2	5	1/100	4×10^{-5}	NACC	$(4, 1 \times 10^4, 0.3, -0.0005, 0.05, 30, 1.85)$
(Figure 5.35) Wrist	20K	1/256	433.2K	0.015	5	1/100	2×10^{-4}	HB	$(100, 2250, 0.125, 10, 1, 10)$

Comparison with Gissler et al. (2020) Our method shares similarities with Gissler et al. (2020) in computing the velocity gradient and advecting the deformation gradient using SPH-based spatial discretization. However, Gissler et al. (2020) utilizes a Jacobi solver for the primal system, whereas we solve the dual formulation with a Lagrangian multiplier, enabling coupling with traditional PBD materials. A comprehensive discussion of the advantages/disadvantages of dual vs. primal formulations can be found at Macklin et al. (2020). Most notably, as shown in Figure 5.28, our approach to plasticity is distinct. We adopt the same snow constitutive model (Stomakhin et al., 2013) for both methods, with parameters: $\rho = 400\,\text{kg/m}^3$, $E = 5 \times 10^5\,\text{Pa}$, hardening coefficient $\xi = 10$, critical compression $\theta_c = 0.025$, and critical stretch $\theta_s = 0.0075$. We conducted simulations using timesteps of $\Delta t = 10^{-4}\,\text{s}$ and $10^{-3}\,\text{s}$. Gissler et al. (2020) employs a semi-implicit plasticity model with a single post-return-mapping projection per time step, akin to Stomakhin et al. (2013), making snow behavior timestep dependent due to this explicit plastic deformation update. In contrast, our method uses a fully implicit plasticity treatment, alternating between XPBD iteration and fixed-point iteration and producing consistent behaviors across different timestep sizes.

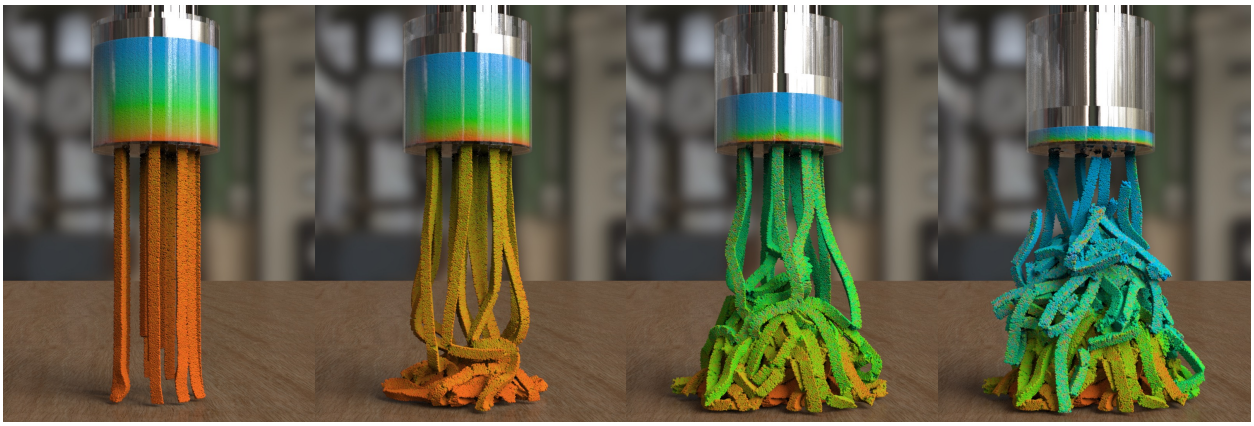


Figure 5.29: **Noodles.** We simulate noodles modeled using Von Mises plasticity as it is pressed through a cylindrical mold.

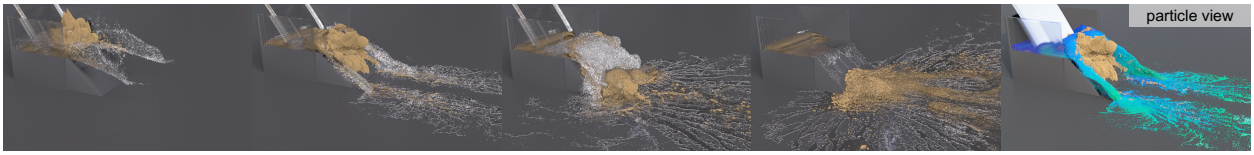


Figure 5.30: **Dam Breach.** Our method can be seamlessly coupled with PBF (Macklin and Müller, 2013) to simulate sand and water mixture.

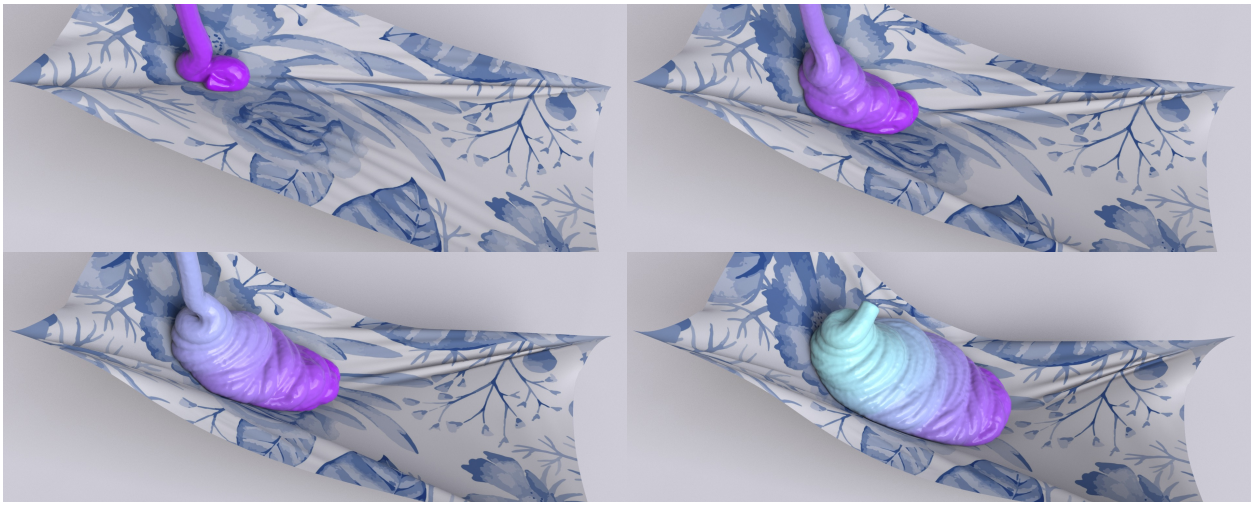


Figure 5.31: **Cloth.** XPBI fits into traditional XPBD pipeline and naturally couples updated Lagrangian materials (viscoplastic paint) and mesh-based geometry (cloth).

5.2.5.2 Demos

Complex materials with up to millions of particles, such as mud (Figure 5.29, Figure 5.30), viscoplastic paint (and its coupling with traditional XPBD cloth) (Figure 5.31), brittle

Figure 5.32: **Candy Camponotus.** We simulate the brittle fracture of a candy shaped like a camponotus falling onto the ground.

Figure 5.33: **Hourglass.** Sand in hourglass accumulates at the bottom. The material is modeled with Drucker-Prager plasticity (Klár et al., 2016).

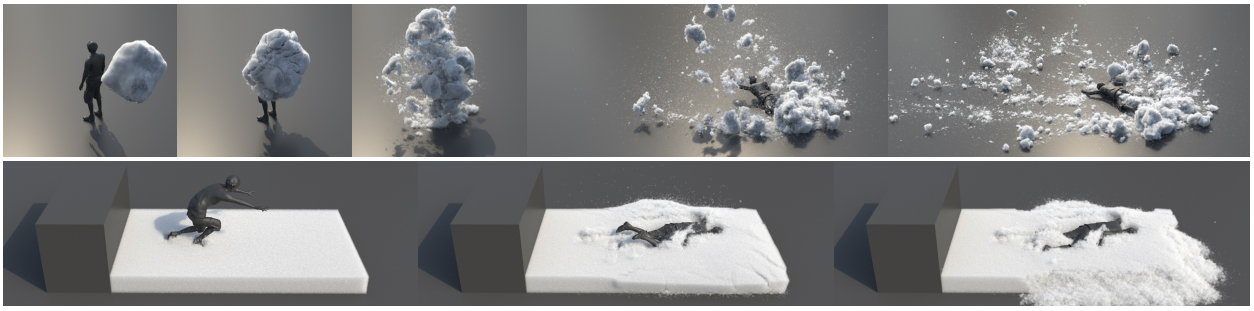


Figure 5.34: **Hitman and Snow Dive.** We successfully reproduce realistic and complex snow behaviors, such as a snowball hitting a person (**top**) and a person falling into a snow ground (**bottom**), using NACC (Wolper et al., 2019) constitutive model.

fracture (Figure 5.32), sand (Figure 5.33), and snow (Figure 5.34) can be simulated with XPBI. The timing and parameters are summarized in Table 5.3.

Real-time Interaction The position-based method family is widely adopted in game and VR applications due to its real-time interactive capabilities (Barreiro et al., 2017; Jiang

Figure 5.35: **Real-time Vision Pro™ Interaction.** Interactive manipulation of a viscoelastic fluid, consisting of up to 20K particles, is simulated at 30fps.

et al., 2024). We further showcase our method in interactive applications where very small time steps are impractical. The convergence and stability of our approach enable interactive performance in moderately complex scenarios involving 20K particles. For this application, we employ the Jacobi solver due to its significant parallelism capabilities on GPUs for small-scale simulation. We use an Apple Vision Pro VR device and *VisionProTeleop* (Park and Agrawal, 2024) to track hand motions and enable a virtual hand to interact with viscous fluids.

5.2.6 Discussion

In summary, XPBI is a novel updated Lagrangian enhancement for XPBD, enhancing its capability for simulating complex inelastic behaviors governed by continuum mechanics-based constitutive laws. Further incorporating an implicit plasticity treatment and stability enhancements, XPBI can be easily integrated into standard XPBD to open up its new simulation possibilities.

Given the high stiffness and detailed resolution of most scenes, the timestep is constrained by the relative low GS convergence and SPH CFL condition, which is related to the kernel's support radius. We note that although our method falls under the category of implicit methods capable of handling highly stiff materials, additional damping models, such as XSPH or XPBD constraint damping, are still necessary to avoid jittering effects or stability issues.

While XPBI supports a broad range of material behaviors, Maxwell viscoelastic materials (Fang et al., 2019) necessitate more specialized treatment. Also, interactions between sand and water mixtures occur primarily at the material boundary. Simulating actual porous media (Tampubolon et al., 2017) and fluid sediment mixture (Gao et al., 2018a) with proper

momentum transfer are interesting future work. Another direction is to optimize parallelism for solving inelastic per-particle constraints. Although grid-colored Gauss-Seidel significantly improves performance over sequential iterations in large-scale simulations, it underperforms in small-scale cases on modern GPUs due to low utilization, obstructing many interactive rate experiments. A tailored real-time solver would be interesting.

CHAPTER 6

Real-to-Sim Tasks

6.1 PAC-NeRF: Physics Augmented Continuum Neural Radiance Fields for Geometry-Agnostic System Identification

6.1.1 Introduction

Inferring the geometric and physical properties of an object directly from visual observations is a long-standing challenge in computer vision and artificial intelligence. Current machine vision systems are unable to disentangle the geometric structure of the scene, the dynamics of moving objects, and the mechanisms underlying the imaging process – an innate cognitive process in human perception. For example, by merely watching someone kneading and rolling dough, we are able to disentangle the dough from background clutter, form a predictive model of its dynamics, and estimate physical properties, such as its consistency to be able to replicate the recipe. There exists a large body of work on inferring the geometric (*extrinsic*) structure of the world from multiple images (e.g., structure-from-motion (Hartley and Zisserman, 2003)). This has been bolstered by recent approaches leveraging differentiable rendering pipelines (Tewari et al., 2022) and neural scene representations (Xie et al., 2022a), unlocking a new level of performance and visual realism. On the other hand, approaches to extract the physical (*intrinsic*) properties (e.g., mass, friction, viscosity) from images are yet nascent (Jatavallabhula et al., 2020; Ma et al., 2021; Jaques et al., 2022, 2020) – all assume full knowledge of the geometric structure of the scene, thereby limiting their applicability.

The key question we ask in this work is "*can we recover both the geometric structure and the physical properties of a wide range of objects from multi-view video sequences*"? This

dispenses with all of the assumptions made by state-of-the-art approaches to video-based system identification (known geometries in (Ma et al., 2021) and additionally rendering configurations in (Jatavallabhula et al., 2020)). Additionally, the best performing approaches to recover geometries (but not physical properties) of dynamic objects in videos include variants of neural radiance fields (NeRF) (Mildenhall et al., 2020), such as (Pumarola et al., 2021a; Tretschk et al., 2021; Park et al., 2021). However, all such neural representations of dynamic scenes need to learn object dynamics from scratch, requiring a significant amount of data to do so, while also being uninterpretable. We instead employ a differentiable physics simulator as a more prescriptive, data-efficient, and generalizable dynamics model; enabling parameter estimation solely from videos.

Our approach—Physics Augmented Continuum Neural Radiance Fields (PAC-NeRF)—is a novel system identification technique that assumes nothing about the geometric structure of a system. PAC-NeRF is extremely general – operating on deformable solids, granular media, plastics, and Newtonian/non-Newtonian fluids. PAC-NeRF brings together the best of both worlds; differentiable physics and neural radiance fields for dynamic scenes. By augmenting a NeRF with a differentiable continuum dynamics model, we obtain a unified model that estimates object geometries *and* their physical properties in a single framework.

Specifically, a PAC-NeRF $\mathcal{F}$ is a NeRF, comprising a volume density field and a color field, coupled with a velocity field $\boldsymbol{v}$ that admits the continuum conservation law: $\frac{D\mathcal{F}}{Dt} = 0$ (Spencer, 2004). In conjunction with a hybrid Eulerian-Lagrangian formulation, this allows us to advect geometry and appearance attributes to all frames in a video sequence, enabling the specification of a reconstruction error in the image space. This error term is minimized by gradient-based optimization, leveraging the differentiability of the entire computation graph, and enables system identification over a wide range of physical systems, where neither the geometry nor the rendering configurations are known. Our hybrid representation considerably speeds up the original MLP-based NeRF by efficient voxel discretization (Sun et al., 2022), and also conveniently handles collisions in continuum simulations, following the MPM pipeline (Jiang et al., 2015a). The joint differentiable rendering-simulation pipeline with a unified Eulerian-Lagrangian conversion is highly optimized for high-performance computing

on GPU.

In summary, we make the following contributions.

- We propose **PAC-NeRF** – a dynamic neural radiance field that satisfies the continuum conservation law (Section 6.1.3.1).
- We introduce a *hybrid Eulerian-Lagrangian representation*, seamlessly blending the Eulerian nature of NeRF with MPM's Lagrangian particle dynamics. (Section 6.1.3.3).
- Our framework estimates *both* the geometric structure and physical parameters of a wide variety of complex systems, including elastic materials, plasticine, sand, and Newtonian/non-Newtonian fluids, outperforming state-of-the-art approaches by up to **two orders of magnitude**. (Section 6.1.5).

6.1.2 Related Work

Neural radiance fields (NeRF), introduced in Mildenhall et al. (2020), are a widely adopted technique to encode scene geometry in a compact neural network; enabling photo-realistic rendering and depth estimation from novel views. A comprehensive survey of neural fields is available in (Xie et al., 2022a). In this work, we adopt the voxel representation proposed by Sun et al. (2022) as these do not require positional information and naturally fit the Eulerian stage of the Material Point Method (MPM) used in our physics prior.

For **perception of dynamic scenes**, Li et al. (2021g) introduce forward and backward motion fields to enforce consistency in the representation space of neighboring frames. D-NeRF (Pumarola et al., 2021a) introduces a canonical frame with a unique neural field for densities and colors, and a time-dependent backward deformation map to query the canonical frame. This representation has since been adopted in (Tretschk et al., 2021) and Park et al. (2021). Chu et al. (2022) targets smoke scenes and advects the density field by the velocity field of smoke. This method does not deal with boundary conditions, so it cannot model solids and contact. Guan et al. (2022) present a combination of NeRF with intuitive fluid dynamics leveraging neural simulators; whereas we provided a principled, and interpretible

simulation-and-rendering framework.

System identification of soft bodies is an extremely challenging task, owing to its high dimensionality and the presence of large deformations. Neural (Sanchez-Gonzalez et al., 2020; Li et al., 2019b; Xu et al., 2019b) or gradient-free methods (Wang et al., 2015; Takahashi and Lin, 2019b) struggle to achieve high accuracy on these problems, owing to their black-box nature. Recent progress in differentiable physics simulation has demonstrated great promise (Qiao et al., 2021a; Du et al., 2021; Rojas et al., 2021; Geilinger et al., 2020; Heiden et al., 2021; Jatavallabhula et al., 2020; Ma et al., 2021), but assumes that watertight mesh of objects are available. Recently, (Chen et al., 2022a; Qiao et al., 2022) estimate elastic object properties from videos. Our hybrid representation and simulation-rendering is general, and speeds up neural rendering by *two orders of magnitude*.

We use the **Material Point Method (MPM)** due to its ability to handle topology changes and frictional contacts; allowing the simulation of a wide range of materials, including elastic objects, sand (Klár et al., 2016), to fluids (Jiang et al., 2015a) and foam (Yue et al., 2015). While MPM has previously been used for differentiable physics simulation (Hu et al., 2020; Huang et al., 2021; Fang et al., 2022), a watertight, non-degenerate mesh model was assumed to be available. Our method solves the long-standing challenge of perceiving geometric and physical properties solely from videos.

6.1.3 Method

Problem specification: Given a set of (posed) multi-view videos of a dynamic scene, we aim to recover (1) an explicit geometric representation, and (2) physical properties (such as Young's modulus, fluid viscosity, friction angles, etc.) of the dynamic object of interest.

Unlike existing system identification methods that operate on images, we do not require known object geometries. Our approach is general and works on a wide range of material types (fluids, sand, plasticine, etc.). Our proposed approach, Physics Augmented Continuum Neural Radiance Fields **PAC-NeRF**, seamlessly blends neural scene representations and explicit differentiable physics engines for continuum materials. The core components of

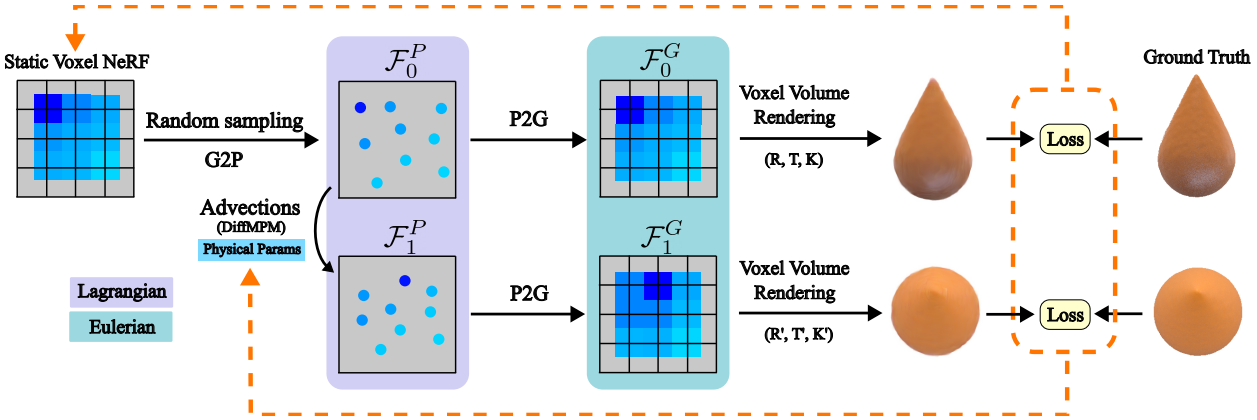


Figure 6.1: **PAC-NeRF** uses both Lagrangian (particle; material-space) and Eulerian (grid; world-space) representations for an accurate yet tractable computational model of continuum materials. P2G and G2P denote particle-to-grid and grid-to-particle transforms, respectively. Renderable quantities (volume densities, colors) are represented in the world space (first frame) using a voxel NeRF (Sun et al., 2022). These quantities are bound to particles (by a sampling scheme) whose dynamics are simulated by using a differentiable material point method (MPM) (Jiang et al., 2015a). The (Eulerian) voxel representation enables efficient collision handling and rendering. Since the entire simulation and rendering pipeline are differentiable, rendered (color) images are able to optimize both the geometric and physical properties of objects. PAC-NeRF (1) accelerates NeRF rendering with Eulerian representation, (2) lends physical plausibility, interpretability, and data efficiency in dynamic scenes, and (3) enables physical parameter estimation for continuum materials.

PAC-NeRF include a continuum NeRF, a particle-grid interconverter, and a Lagrangian field. We detail these in this section (see Figure 6.1).

6.1.3.1 Continuum Neural Radiance Fields

Recall that a (static) NeRF comprises a view-independent volume density field $\sigma(\mathbf{x})$ and a view-dependent appearance (color) field $\mathbf{c}(\mathbf{x}, \omega)$ for each point $\mathbf{x} \in \mathbb{R}^3$, and directions $\omega = (\theta, \phi) \in \mathbb{S}^2$ (spherical coordinates). A *dynamic* (time-dependent) NeRF extends the fields above with an additional time variable $t \in \mathbb{R}^+$, denoted $\sigma(\mathbf{x}, t)$ and $\mathbf{c}(\mathbf{x}, \omega, t)$ respectively. We use the efficient voxel discretization from (Sun et al., 2022) to specify a dynamic Eulerian (world-frame) NeRF. Color images are rendered from the above time-dependent fields by sampling points along a ray, for each pixel in the resultant image. The appearance $\mathbf{C}(\mathbf{r}, t)$ of a pixel specified by ray direction $\mathbf{r}(s)$ ($s \in [s_{\min}, s_{\max}]$) is given by the volume rendering integral (Mildenhall et al., 2020)

$$\boldsymbol{C}(\boldsymbol{r},t)=\int_{s_n}^{s_f} T(s,t)\ \sigma(\boldsymbol{r}(s),t)\ \boldsymbol{c}(\boldsymbol{r}(s),\omega,t)\ ds+\boldsymbol{c}_{bg}\ T(s_f,t),\quad T(s,t)=\exp\left(-\int_{s_n}^{s}\sigma(\boldsymbol{r}(\bar{s}),t)\ d\bar{s}\right) \tag{6.1}$$

The dynamic NeRF can be trained by enforcing the rendered pixel colors to match those in the video.

$$\mathcal{L}_{render}=\frac{1}{N}\sum_{i=0}^{N-1}\frac{1}{|\mathcal{R}|}\sum_{\boldsymbol{r}\in\mathcal{R}}\|\boldsymbol{C}(\boldsymbol{r},t_i)-\hat{\boldsymbol{C}}(\boldsymbol{r},t_i)\|^2, \tag{6.2}$$

where N is the number of frames of videos, $\hat{\boldsymbol{C}}(\boldsymbol{r},t)$ is the ground truth color observation.

Additionally, we enforce that the appearance and volume density fields admit conservation laws characterized by the velocity field of the underlying **physical** system:

$$\frac{D\sigma}{Dt}=0,\quad \frac{D\boldsymbol{c}}{Dt}=0, \tag{6.3}$$

with $\frac{D\phi}{Dt}=\frac{\partial\phi}{\partial t}+\boldsymbol{v}\cdot\nabla\phi$ being the material derivative of an arbitrary time-dependent field $\phi(\boldsymbol{x},t)$. Here, $\boldsymbol{v}$ is a *velocity field*, which in turn must obey momentum conservation for continuum materials

$$\rho\frac{D\boldsymbol{v}}{Dt}=\nabla\cdot\boldsymbol{T}+\rho\boldsymbol{g}, \tag{6.4}$$

where ρ is the physical density field, $\boldsymbol{T}$ is the internal Cauchy stress tensor, and $\boldsymbol{g}$ is the acceleration due to gravity. We use the differentiable Material Point Method (Hu et al., 2020) to evolve Equation (6.4).

6.1.3.2 Particle-Grid Interconverters

While a Lagrangian representation is ideal for advection by the material point method (MPM), an Eulerian frame is required for rendering the advected particle states to image space. We therefore employ a hybrid representation to blend the best of both worlds. A key requirement is to be able to seamlessly traverse the Eulerian (grid) view to the Lagrangian (particle) view (and vice versa).

Denoting the Eulerian and Lagrangian views G and P respectively, a field $\mathcal{F}_*^G(t)=$

$\{\sigma(\mathbf{x}, t), \mathbf{c}(\mathbf{x}, t)\}$ at time t may be interconverted as follows:

$$\mathcal{F}_p^P \approx \sum_i w_{ip} \mathcal{F}_i^G, \quad \mathcal{F}_i^G \approx \frac{\sum_p w_{ip} \mathcal{F}_p^P}{\sum_p w_{ip}}, \tag{6.5}$$

where i indices grid nodes and p indices particles, and w_{ip} is the weight of the trilinear shape function defined on node i and evaluated at the location of particle p. We use *P2G* and *G2P* to denote the particle-to-grid and grid-to-particle conversion processes respectively.

6.1.3.3 Lagrangian field

An Eulerian voxel field $\mathcal{F}^G(t_0)$ is initialized over the first frame of the sequence. From this field, we use the *G2P* process to obtain a Lagrangian particle field $\mathcal{F}^P(t_0)$. We advect this field using an initial guess physical parameter set $\boldsymbol{\Theta}$ and the material point method (Equation (6.3)) to obtain $\mathcal{F}^P(t_1)$ at $t_1 = t_0 + \delta t$ (where δt is the duration of each simulation timestep). The advected field is then mapped to the Eulerian view using the *P2G* process, resulting in $\mathcal{F}^G(t_1)$, which is employed for collision handling and neural rendering. The Eulerian voxel grid representation used here is at least two orders of magnitude ($100\times$) faster in terms of image rendering time (Sun et al., 2022).

Following (Sun et al., 2022), the rendering density field σ and color field $\boldsymbol{c}$ at time t are:

$$\sigma(\boldsymbol{x}, t) = \text{softplus}(\text{Interp}(\boldsymbol{x}, \hat{\sigma})), \quad \boldsymbol{c}(\boldsymbol{x}, \boldsymbol{d}, t) = \text{MLP}(\text{Interp}(\boldsymbol{x}, \hat{\boldsymbol{c}}), \boldsymbol{d}), \tag{6.6}$$

where $\hat{\sigma}$ is a scalar field and $\hat{\boldsymbol{c}}$ is a vector field, both are discretized on a fixed voxel grid. Interp$(\cdot)$ denotes trilinear interpolation. Advection is performed by first advecting the grids $\hat{\sigma}$ and $\hat{\boldsymbol{c}}$, followed by computing the interpolation functions and evaluating the MLP (or the softplus).

The initialization of forward simulation requires generating Lagrangian representations of $\hat{\sigma}$ and $\hat{\boldsymbol{c}}$. We randomly sample 8 particles within each voxel grid and use Equation (6.5) to bind the density and color values onto particles. Additionally, we associate with each particle a scalar value $\alpha_p = 1 - e^{-\text{softplus}(\hat{\sigma}_p)}$ in $(0, 1)$. A lower α value denotes a smaller contribution

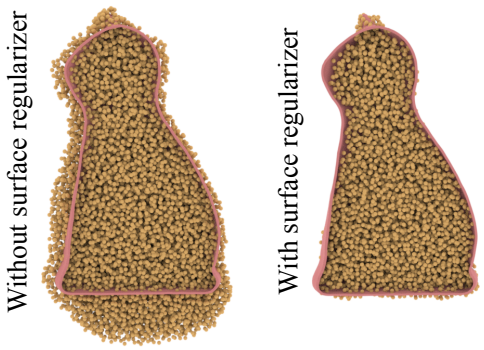


Figure 6.2: Our surface regularizer improves reconstruction quality, by producing a tight-fit shape to the segmentation mask.

to the radiance fields. We make particles with lower values of α softer by scaling the physical density field ρ (Equation (6.4)) and the physical stress field $\boldsymbol{T}$ by a factor α^3. We remove a particle p if $\alpha_p < \epsilon \ \max_p \ \alpha_p$, where we set $\epsilon = 10^{-3}$ as a constant threshold.

6.1.3.4 PAC-NeRF for geometry-agnostic system identification

Our pipeline for geometry-agnostic system identification comprises three distinct phases for computational tractability. We first preprocess data using video matting techniques to extract foreground objects of interest. This is followed by a geometry seeding phase, where we employ a coarse-to-fine approach to recover object geometry. The extracted geometry is then used to perform system identification by rendering out future video frames based on a guessed set of physical properties, computing an error term with respect to the true videos, and updating the physical properties by gradient-based optimization.

Data Preprocessing: We assume a set of static cameras with known intrinsics and extrinsics. To focus rendering computation on the object of interest, we run the video matting framework in Lin et al. (2021) to remove static background objects. This also provides us with a segmentation mask of the foreground object of interest.

Geometry Seeding: We first obtain a coarse geometry of the foreground object(s) of interest by employing the static voxel fields. The contents of the foreground segmentation mask are rendered to produce an appearance loss term optimized using gradient descent. We

employ a coarse-to-fine strategy to enable easier optimization. We noticed that, as opposed to directly rendering the initial voxel radiance field, running the G2P converter followed by a P2G converter ensures that rendering for all the frames (including the initial frame) is consistently based on the same group of particles (implicitly providing a set of consistent correspondences across time). In addition to the rendering loss Equation (6.2), we employ a surface regularizer to regularize the geometric density field:

$$\mathcal{L}_{\text{surf}} = \sum_{p} \text{clamp}(\alpha_p, 10^{-4}, 10^{-1})(\frac{\Delta x}{2})^2. \tag{6.7}$$

This regularizer minimizes the total surface area, and as shown in Figure 6.2, this tends to improve the quality of reconstructed geometries by making the reconstructed point cloud more compact and fit the ground truth boundary more closely.

System Identification: Upon convergence of the geometry seeding phase, we freeze the parameters of the field (for the initial time step t_0). To minimize scenarios where one or more variables of interest become unobservable under unknown initial conditions, we use the first 2-3 frames to estimate the initial velocity of observed particles. We then optimize for physical parameters with each subsequent frame. To mitigate convergence issues in parameter spaces with larger degrees of freedom, we found it helpful to warm start the optimizer after initial collision events, followed by an optimization over the entire sequence.

6.1.4 Implementation Details

The architecture of voxel discretization of NeRF follows Sun et al. (2022), which stores density value and color feature within 160^3 voxels, only contraining an extra 2-layer MLP with a hidden dimension 128 for view-dependent color fields. The dimension of color feature on each voxel is 12. Positional embedding is applied to the inputs (query position, view direction and color feature) to the shallow MLP, leading to a input dimension 39. Our differentiable MPM is using DiffTaichi (Hu et al., 2020). Both the rendering and simulation are optimized for GPU. The entire training takes ~ 1.5 hours on a single Nvidia 3090 GPU. Simulation+rendering of one frame takes $\sim 1s$ (vs. $\sim 10min$ for (Chen et al., 2022a)).

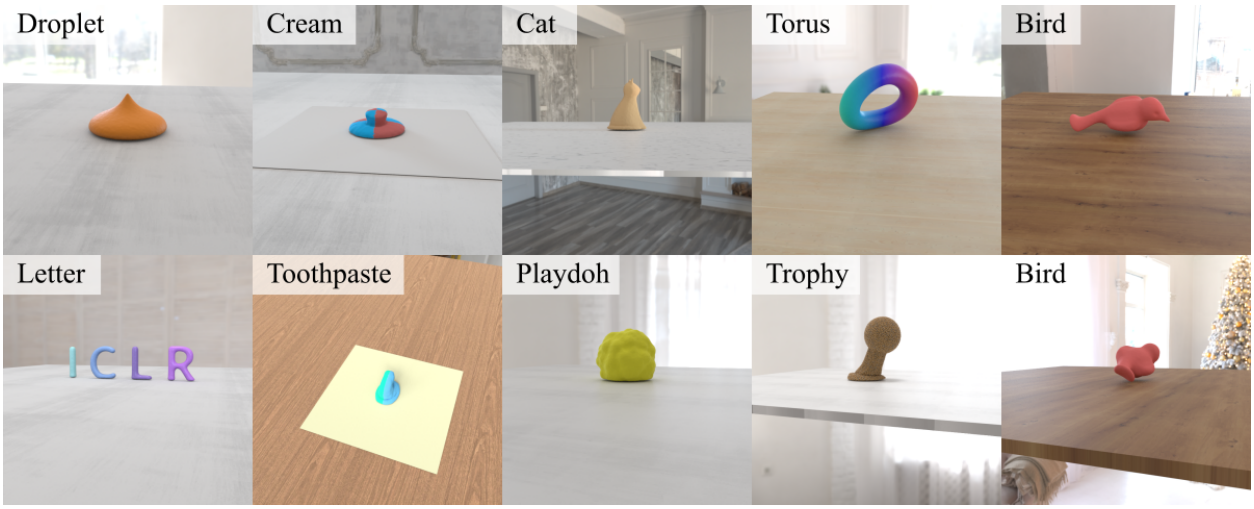


Figure 6.3: The **photorealistic dataset** used to evaluate system identification and geometry estimation. The dataset includes a variety of continuum materials, including Newtonian fluids (Droplet, Letter), non-Newtonian fluids (Cream, Toothpaste), granular media (Trophy), deformable solids (Torus, Bird), and plasticine (Cat, Playdoh). All objects freely fall under the influence of gravity, undergoing collisions. Objects are rendered under complex environmental lighting conditions for photorealism.

6.1.5 Experiments

We conduct various experiments to study the efficacy of PAC-NeRF on system identification tasks and find that:

- PAC-NeRF can recover high-quality object geometries solely from videos.
- PAC-NeRF performs significantly better on system identification tasks compared to fully learned approaches.
- PAC-NeRF alleviates the assumptions that other techniques require (i.e., known object geometry), while outperforming them.
- Purely pixel-based loss functions provide rich gradients that enable physical parameter estimation.

6.1.5.1 Experiment setup

Dataset: To evaluate different system identification methods, we simulate and render a wide range of objects using a photo-realstic simulation engine with varying environment lighting conditions and ground textures. Our dataset includes deformable objects, plastics, granular media, Newtonian and non-Newtonian fluids. Figure 6.3 demonstrates a few example scenarios. In each scene, objects freely fall under the influence of gravity, undergoing (self and external) collisions. For convenience, we assume that the collision objects such as the ground plane are known (however, this can also be easily estimated from observed images). Each scene is captured from 11 uniformly sampled viewpoints, with the cameras evenly spaced on the upper hemisphere containing the object. All ground truth simulation data are generated by MLS-MPM framework (Hu et al., 2018a).

Physical Parameters: Our differentiable MPM implementation supports five kinds of common materials, including elasticity, plasticine, granular media (e.g., sand), Newtonian fluids and non-Newtonian fluids.

- *Elasticity*: Young's modulus (E) (material stiffness), Poisson's ration (ν) (ability to preserve volume under deformation).
- *Plasticine*: Young's modulus (E), Poisson's ration (ν), and yield stress (τ_Y) (stress required to cause permanent deformation/yielding).
- *Newtonian fluid*: fluid viscosity (μ) (opposition to velocity change), bulk modulus (κ) (ability to preserve volume).
- *Non-Newtonian fluid*: shear modulus (μ), bulk modulus (κ), yield stress (τ_Y), and plasticity viscosity (η) (decayed temporary resistance to yielding).
- *Sand*: friction angle (θ_{fric}) (proportionality constant determining slope of a sand pile).

System Identification Pipeline: We first train a static voxel NeRF using data from the first frame (following Sun et al. (2022)) with the Adam optimizer. The initial velocity estimator uses L-BFGS, which we experimentally find to be better than Adam for this sub-task. For all other physical parameters of interest, we use the Adam optimizer.

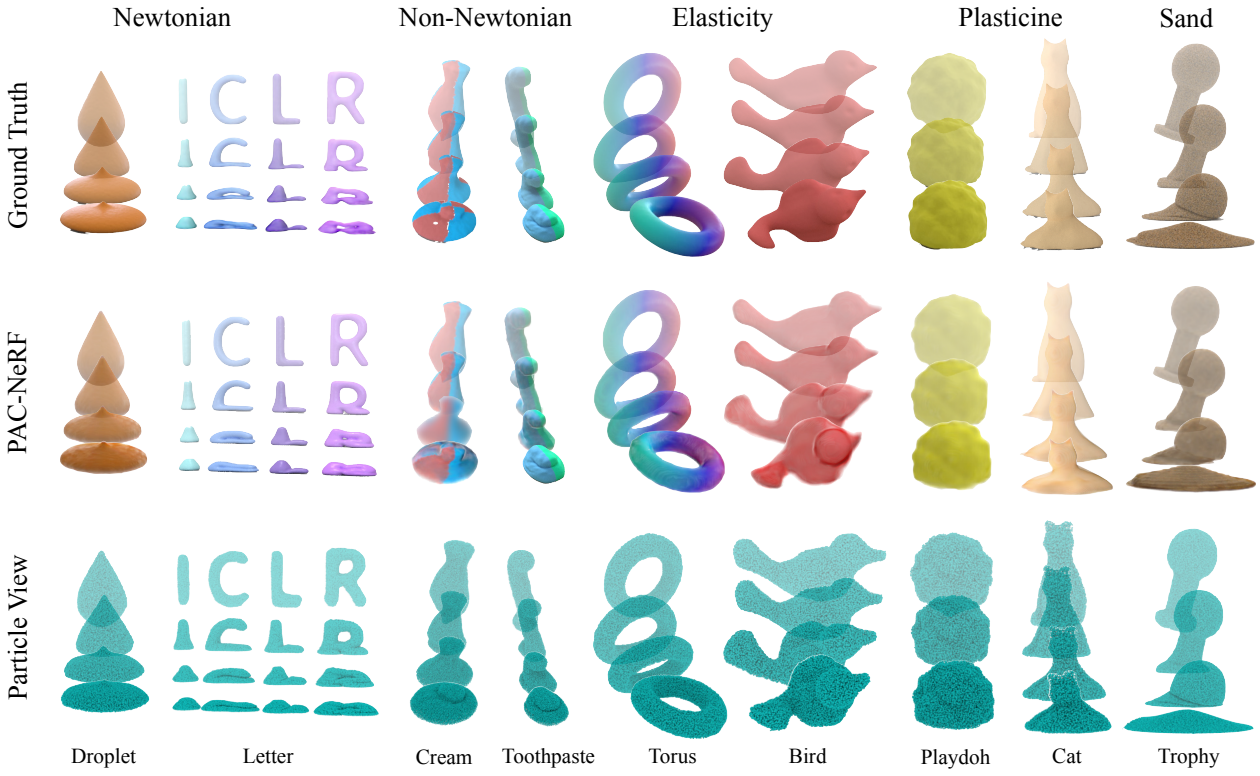


Figure 6.4: **Qualitative results**: Our experiments cover a wide range of continuums, from Newtonian fluids and non-Newtonian fluids, to elastic/plastic solids and granular media (sand). Existing approaches (including dynamic NeRF and variants) are unable to reconstruct these highly dynamic objects. PAC-NeRF not only reconstructs them with high accuracy but also precisely determines the underlying physical properties.

6.1.5.2 System Identification Evaluation

Synthetic Data: We report system identification results using PAC-NeRF over 9 synthetic problem instances. The qualitative results are shown in Figure 6.4. And the quantitative results are listed in Table 6.1. Each row lists the initial guess, the values obtained after optimization, and the ground truth. We see that, in each case, the optimized physical parameters closely agree with the ground truth.

Real Data: We also evaluate PAC-NeRF on real-world data (Figure 6.5). We built a capture system comprising four synchronized Intel RealSense D455 cameras capable of streaming RGB images at a resolution of 640×480 and at a rate of 60 frames per second. Note that we DO NOT use any depth data recorded from these cameras in our experiments, for fair evaluation. We captured a deformable ball falling onto a table and manually segmented

Table 6.1: PAC-NeRF estimates physical parameters very accurately.

	Initial Guess	Optimized	Ground Truth
Droplet	$\mu = 1$, $\kappa = 10^3$	$\mu = 2.09 \times 10^2$, $\kappa = 1.08 \times 10^5$	$\mu = 200$, $\kappa = 10^5$
Letter	$\mu = 1$, $\kappa = 10^6$	$\mu = 83.85$, $\kappa = 1.35 \times 10^5$	$\mu = 100$, $\kappa = 10^5$
Cream	$\mu = 10^2$, $\kappa = 10^4$, $\tau_Y = 10$, $\eta = 1$	$\mu = 1.21 \times 10^5$, $\kappa = 1.57 \times 10^6$, $\tau_Y = 3.16 \times 10^3$, $\eta = 5.6$	$\mu = 10^4$, $\kappa = 10^6$, $\tau_Y = 3 \times 10^3$, $\eta = 10$
Toothpaste	$\mu = 10^2$, $\kappa = 10^4$, $\tau_Y = 10$, $\eta = 1$	$\mu = 6.51 \times 10^3$, $\kappa = 2.22 \times 10^5$, $\tau_Y = 228$, $\eta = 9.77$	$\mu = 5 \times 10^3$, $\kappa = 10^5$, $\tau_Y = 200$, $\eta = 10$
Torus	$E = 10^5$, $\nu = 0.1$	$E = 1.04 \times 10^6$, $\nu = 0.322$	$E = 10^6$, $\nu = 0.3$
Bird	$E = 10^3$, $\nu = 0.1$	$E = 2.78 \times 10^5$, $\nu = 0.273$	$E = 3 \times 10^5$, $\nu = 0.3$
Playdoh	$E = 10^5$, $\nu = 0.4$, $\tau_Y = 10^3$	$E = 3.84 \times 10^6$, $\nu = 0.272$, $\tau_Y = 1.69 \times 10^4$	$E = 2 \times 10^6$, $\nu = 0.3$, $\tau_Y = 1.54 \times 10^4$
Cat	$E = 10^3$, $\nu = 0.2$, $\tau_Y = 10^2$	$E = 1.61 \times 10^5$, $\nu = 0.293$, $\tau_Y = 3.57 \times 10^3$	$E = 10^6$, $\nu = 0.3$, $\tau_Y = 3.85 \times 10^3$
Trophy	$\theta^0_{fric} = 10°$	$\theta^0_{fric} = 36.1°$	$\theta^0_{fric} = 40°$

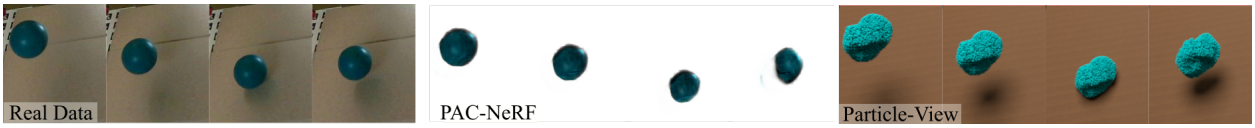


Figure 6.5: We test PAC-NeRF on a set of real-world multiview videos. Our reconstructed scene qualitatively matches the target videos, which validates that the rendering loss is effectively minimized.

the data. Our reconstructed scene qualitatively matches the target videos. We note that, due to the limited number of views (here, 4) we could acquire, there are slight errors in surface geometry estimation. These issues may be mitigated by increasing the number of views from which data is captured.

6.1.5.3 Comparisons

Since there are no existing approaches that estimate *both* geometry and material properties from videos, we make a best-effort comparison by blending combinations of state-of-the-art approaches for each sub-task. For each material type discussed above, we generate 10 problem instances by varying object orientations, initial velocities, and physical parameters. The comparisons are conducted on this dataset. Note that our approach (and most of the baselines herein) does not require a separate training phase – they perform inference-time optimization on each test sequence.

Approaches Evaluated

- **Multi-view LSTM**: To evaluate amortized physical parameter inference schemes, we use a pre-trained ResNet feature extractor to extract features from all views and feed

them into a 2-layer LSTM. This baseline uses privileged information in the form of training video sequences (while all other baselines begin with a random initialization on each test sequence).

- **D-NeRF + DiffSim**: To assess the impact of conservation law enforcement in PAC-NeRF, we compare with VEO (Chen et al., 2022a) (a best-effort comparison) by implementing D-NeRF (dynamic NeRF) and integrating it with our differentiable simulator. A dynamic voxel NeRF learns both the forward and backward deformation field for each frame. The learned forward deformation field is then used to provide the differentiable simulator with 3D supervision.

- **NeRF + ∇Sim**: We also compare with ∇Sim – a state-of-the-art approach for estimating physical properties with both the geometry and rendering configuration known. (By contrast, we assume neither.) We implement a best-effort comparison where we use a static voxel NeRF to recover the geometry and directly provide the rendering color configuration. Note that ∇Sim *only supports FEM simulation* for continuum materials. Therefore, this baseline only runs on a small subset of our scenarios. Moreover, to run this baseline, we extract a surface mesh from our reconstructed point cloud from voxel NeRF and then use TetWild (Hu et al., 2018b) to generate a tetrahedral mesh. Further, we manually choose an optimal camera viewing direction to facilitate optimization.

- **Random**: We also evaluate the performance of these approaches against random chance, by employing a baseline that predicts a uniformly randomly chosen value over the parameter ranges used.

- **Oracle**: We implement an oracle that uses privileged 3D geometric information. The oracle knows the ground truth point clouds, and employs 3D supervision (Chamfer distance) to infer physical properties. Of the 10 problem instances for each material type, we run the oracle on 4.

		Ours	Multi-view LSTM	D-NeRF + DiffSim	NeRF + ∇Sim	Random	Oracle ($\times 10^{-2}$)
Newtonian	$\log_{10}(\mu)$	$\mathbf{11.6 \pm 6.60}$	14.1 ± 9.26	> 291.3	Not supported	64.1 ± 31.3	19.4 ± 6.95
	$\log_{10}(\kappa)$	$\mathbf{16.7 \pm 5.37}$	28.2 ± 19.8	> 85.2		72.8 ± 42.5	258.9 ± 25.2
	v	$\mathbf{0.86 \pm 1.45}$	23.1 ± 17.8	25.5		55.5 ± 22.6	2.82 ± 1.72
Non-Newtonian	$\log_{10}(\mu)$	$\mathbf{24.1 \pm 21.9}$	39.4 ± 26.9	> 276.7	Not supported	34.2 ± 19.9	$\mathbf{24.0 \pm 17.1}$
	$\log_{10}(\kappa)$	44.0 ± 26.3	$\mathbf{25.1 \pm 22.6}$	> 262.5		67.6 ± 49.4	82.90 ± 21.0
	$\log_{10}(\tau_Y)$	$\mathbf{5.09 \pm 7.41}$	7.19 ± 7.88	> 359.6		30.0 ± 15.0	9.75 ± 11.3
	$\log_{10}(\eta)$	$\mathbf{28.7 \pm 23.3}$	39.9 ± 21.1	> 86.1		64.3 ± 43.7	90.6 ± 37.2
	v	$\mathbf{0.29 \pm 0.13}$	24.4 ± 14.2	25.9		52.2 ± 21.6	2.60 ± 0.39
Elasticity	$\log_{10}(E)$	$\mathbf{3.02 \pm 3.72}$	17.7 ± 9.25	> 437.5	151.6 ± 42.6	96.2 ± 46.9	4.39 ± 4.54
	ν	$\mathbf{4.35 \pm 5.08}$	81.8 ± 58.4	7.67	16.4 ± 6.58	10.9 ± 6.37	3.65 ± 2.88
	v	$\mathbf{0.50 \pm 0.23}$	6.10 ± 3.27	30.7	182.4 ± 74.1	43.6 ± 25.4	2.69 ± 0.97
Plasticine	$\log_{10}(E)$	83.8 ± 68.4	41.1 ± 31.4	**23.2**	Not supported	42.9 ± 38.0	56.2 ± 30.7
	$\log_{10}(\tau_Y)$	$\mathbf{11.2 \pm 14.5}$	28.6 ± 17.3	> 268.9		82.7 ± 43.1	8.13 ± 3.61
	ν	18.9 ± 15.7	$\mathbf{5.40 \pm 3.49}$	10.36		10.6 ± 6.15	$\mathbf{4.44 \pm 3.61}$
	v	$\mathbf{0.56 \pm 0.17}$	22.0 ± 14.8	> 384.6		47.7 ± 17.53	4.06 ± 1.87
Sand	θ_{fric}	$\mathbf{4.89 \pm 1.10}$	20.1 ± 2.06	64.5	Not supported	22.5 ± 14.7	$\mathbf{0.44 \pm 0.42}$
	v	$\mathbf{0.21 \pm 0.08}$	14.6 ± 6.25	40.6		38.7 ± 21.1	1.17 ± 0.42

Table 6.2: **System identification performance**: Means and the standard derivations of absolute errors are reported for each metric. We compare with five baselines, including a pure vision-based method (ResNet+LSTM regression), D-NeRF+DIffSim (similar to VEO (Chen et al., 2022a)), NeRF+∇Sim (Jatavallabhula et al., 2020), random sampling from parameter distribution, and oracle point cloud plus Chamfer distance minimization. Our method obtains the best results (highlighted in boldface font) in 14/17 entries, excluding the Oracle.

Analysis of Results The performances of all approaches on the system identification tasks are listed in Table 6.2. For each material, we report the mean absolute error and its standard deviation over the evaluated instances. We see that compared to all other methods, PAC-NeRF achieves the best results in 14 of the 17 categories of physical properties estimated.

The multi-view LSTM method is adapted from DensePhysNet (Xu et al., 2019b) and learns the mapping from videos to physical parameters implicitly. Notably, this baseline requires privileged information in the form of training sequences – not available to the other approaches.

The D-NeRF+Diffsim is adapted from VEO (Chen et al., 2022a), where the forward deformation field is used to optimize a differentiable simulation. The accuracy of this approach therefore relies on the quality of the learned forward deformation field. However, this learned deformation field (Tretschk et al., 2021) cannot guarantee physical correctness, unlike PAC-NeRF which is constrained by the conservation laws (Equation (6.3) and Equation (6.4)). In

Table 6.3: Quantitative results of elastic rope example.

	Initial Guess	Optimized	Ground Truth
Rope (Short)	$E = 10^3$, $\nu = 0.4$	$E = 1.12 \times 10^5$, $\nu = 0.22$	$E = 10^5$, $\nu = 0.3$
Rope (Long)	$E = 10^4$, $\nu = 0.4$	$E = 1.09 \times 10^6$, $\nu = 0.31$	$E = 10^6$, $\nu = 0.3$

our experiments, we observed that this leads to very noisy deformation, and impedes system identification performance. Due to these instabilities of this approach, we only run this on one scenario per material type and report its performance.

The NeRF+∇Sim (Jatavallabhula et al., 2020) baseline only supports FEM simulations for elastic materials and is sensitive to time integration step size. To stabilize the symplectic FEM and the contact model used in ∇Sim, tens of thousands of substeps are required to simulate the entire sequence. Notwithstanding the errors induced due to geometric inaccuracies and unknown rendering configurations, errors accumulated from long timestepping sequences also contribute to the failure of ∇Sim on our scenarios. PAC-NeRF, with its Eulerian-Lagrangian representation, is more robust under larger deformations (e.g., fluids and sand) and permits larger time step sizes than a FEM simulation with a symplectic integrator (used in ∇Sim).

The Oracle baseline assumes known 3D (point cloud) geometry and computes a Chamfer distance error term per-timestep. We note, surprisingly, that in several scenarios, a pixel-wise color loss outperforms this 3D supervision signal. The results show the effectiveness of the differentiable rendering-simulation pipeline, where a 2D pixel-level supervision better optimizes the physical parameters, opposed to a 3D Chamfer distance metric.

6.1.5.4 Additional Evaluations

Complex Boundary Conditions Pre-known boundary conditions are trivially to be added in MPM simulations, just like the ground. Here we shown examples of elastic rope falling onto two rigid cylinders in Figure 6.6. The initial guess, optimized values and the ground truth values of physical parameters are listed in Table 6.3.

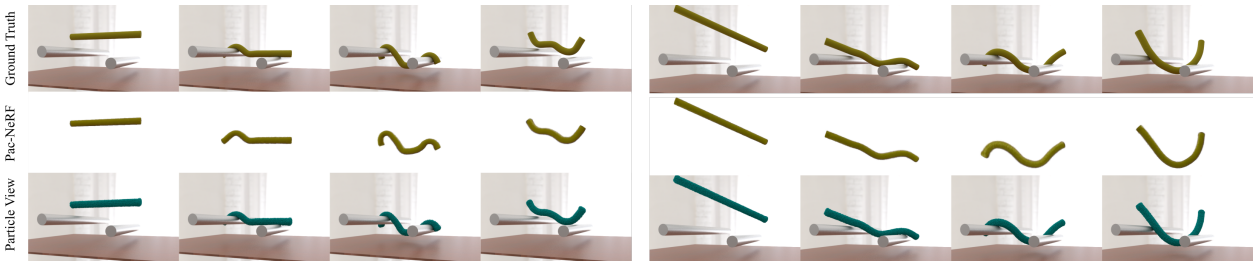


Figure 6.6: Examples of elastic ropes falling onto cylinders. The top row is the ground truth video. PAC-NeRF is trained on the segmented data, whose rendering outputs are visualized in the middle row. The bottom row is the reconstructed MPM particles.

Figure 6.7: D-NeRF suffers from major artifacts as the object undergoes sudden large deformations. Notice how the object becomes partially absent in the middle two frames.

Reconstruction Quality Comparison with D-NeRF When evaluating the D-NeRF+Diffsim baseline, we observe that D-NeRF is sensitive to sudden, large deformations (see Figure 6.7). As the simulation progresses, D-NeRF suffers from artificial extreme deformation, resulting in the object breaking apart. This degrades the simulation to the extent that parts of the object even partially disappear (violating physical plausibility). This artifact is due to the learned backward deformation map points to an empty area in the canonical space, resulting in zero volume density. By contrast, our PAC-NeRF constrains forward deformations to follow the physical conservation law and does not experience this kind of unphysical result.

Qualitative Comparison with ∇Sim on Real-World Data We also tried ∇Sim on our real-world data. The qualitative comparison between PAC-NeRF and ∇Sim is shown in Figure 6.8. As we do in the baseline comparisons, we extract a surface mesh from our reconstructed point cloud from voxel NeRF and then generate a tetrahedral mesh, manually pick an optimal camera and set an approximate rendering configuration. As we discussed in our baseline comparison, explicit FEM used in ∇Sim suffers from tiny time steps, leading to significant numerical errors in backpropagations. Combined with errors from the geometry

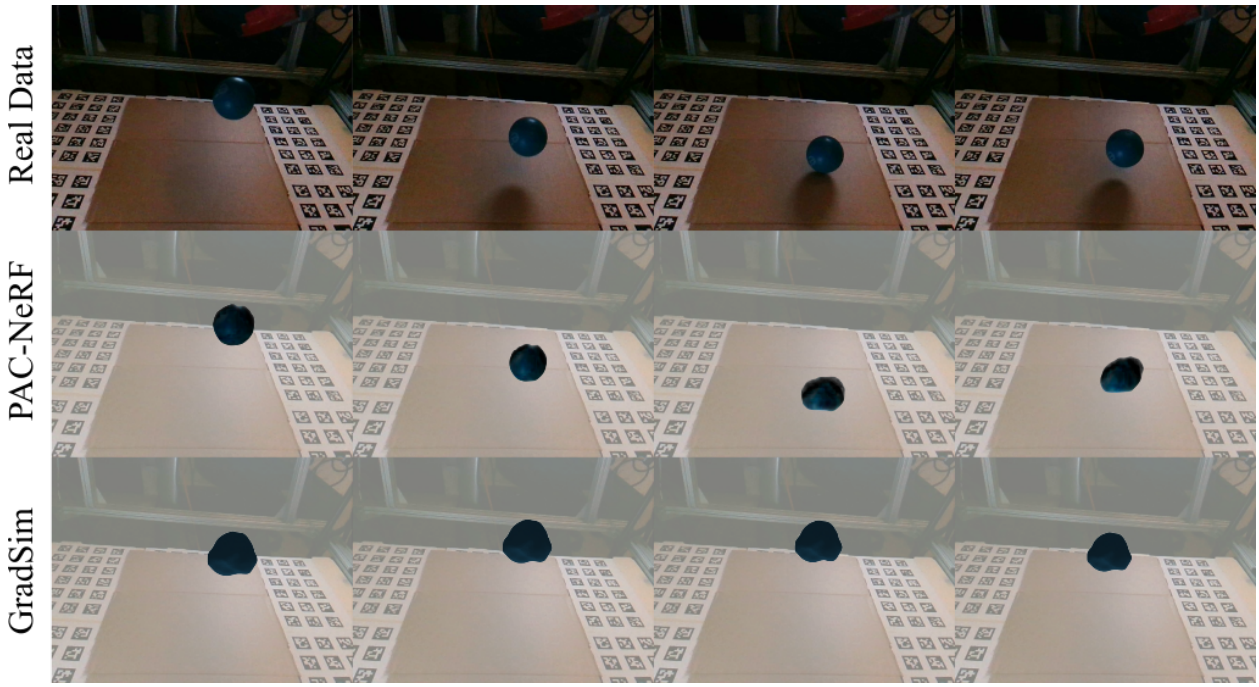


Figure 6.8: Qualitative comparison with ∇Sim on real-world data.

reconstructions and the approximate rendering configurations, all these factors contribute to the failure of ∇Sim.

6.1.6 Conclusion

We presented Physics Augmented Continuum Neural Radiance Fields (PAC-NeRF), a novel representation for system identification without geometry priors. The continuum reformulation of dynamic NeRF naturally unifies physical simulation and rendering, enabling differentiable simulators to estimate both geometry and physical properties from image space signals (such as multi-view videos). A hybrid representation is used to discretize our PAC-NeRF implementation, which leverages the high efficiency of an Eulerian voxel-based NeRF and the flexibility of MPM.

Despite the promising results, this work has some limitations. It assumes the availability of synchronized and accurately calibrated cameras to ensure good-quality reconstruction using NeRF. The scenes used in this work should also be easily amenable to video matting (background removal). Moreover, this work assumes the underlying physical phenomenon

follows continuum mechanics and cannot automatically distinguish between different materials. Future work could focus on extending the MPM framework beyond volumetric continuum materials, such as cloth, using neural constitutive models (Li et al., 2022d), implicit MPM for stiff materials, and integrating other differentiable simulators like articulated body simulators. Additionally, incorporating interactions with rigid bodies could enable manipulation tasks (Huang et al., 2021) on NeRF-represented soft-bodies.

6.2 PhysGaussian: Physics-Integrated 3D Gaussians for Generative Dynamics

6.2.1 Introduction

Recent strides in Neural Radiance Fields (NeRFs) have showcased significant advancements in 3D graphics and vision (Mildenhall et al., 2021). Such gains have been further augmented by the cutting-edge 3D Gaussian Splatting (GS) framework (Kerbl et al., 2023). Despite many achievements, a noticeable gap remains in the application towards generating novel dynamics. While there exist endeavors that generate new poses for NeRFs, they typically cater to quasi-static geometry shape editing tasks and often require meshing or embedding visual geometry in coarse proxy meshes such as tetrahedra (Yuan et al., 2022; Xu and Harada, 2022; Peng et al., 2022; Jambon et al., 2023).

Meanwhile, the traditional physics-based visual content generation pipeline has been a tedious multi-stage process: constructing the geometry, making it simulation-ready (often through techniques like tetrahedralization), simulating it with physics, and finally rendering the scene. This sequence, while effective, introduces intermediary stages that can lead to discrepancies between simulation and final visualization. Even within the NeRF paradigm, a similar trend is observed, as the rendering geometry is embedded into a simulation geometry. This division, in essence, contrasts with the natural world, where the physical behavior and visual appearance of materials are intrinsically intertwined. Our overarching philosophy seeks to align these two facets by advocating for a unified representation of a material substance,

Figure 6.9: **PhysGaussian** is a unified simulation-rendering pipeline based on 3D Gaussians and continuum mechanics.

employed for both simulation and rendering. In essence, our approach champions the principle of *"what you see is what you simulate" (WS^2)* (Müller et al., 2016), aiming for a more coherent integration of simulation, capturing, and rendering.

Building towards this goal, we introduce PhysGaussian: physics-integrated 3D Gaussians for generative dynamics. This novel approach empowers 3D Gaussians to encapsulate physically sound Newtonian dynamics, including realistic behaviors and inertia effects inherent in solid materials. More specifically, we impart physics to 3D Gaussian kernels, endowing them with kinematic attributes such as velocity and strain, along with mechanical properties like elastic energy, stress, and plasticity. Notably, through continuum mechanics principles and a custom Material Point Method (MPM), PhysGaussian ensures that both physical simulation and visual rendering are driven by 3D Gaussians. This eradicates the necessity for any embedding mechanisms, thus eliminating any disparity or resolution mismatch between the simulated and the rendered.

We present PhysGaussian's versatile adeptness in synthesizing generative dynamics across various materials, such as elastic objects, metals, non-Newtonian viscoplastic substances (e.g. foam or gel), and granular mediums (e.g. sand or soil). To summarize, our contributions include

- **Continuum Mechanics for 3D Gaussian Kinematics:** We introduce a continuum mechanics-based strategy tailored for evolving 3D Gaussian kernels and their associated spherical harmonics in physical Partial Differential Equation (PDE)-driven displacement

fields.

- **Unified Simulation-Rendering Pipeline:** We present an efficient simulation and rendering pipeline with a unified 3D Gaussian representation. Eliminating the extra effort for explicit object meshing, the motion generation process is significantly simplified.

- **Versatile Benchmarking and Experiments:** We conduct a comprehensive suite of benchmarks and experiments across various materials. Enhanced by real-time GS rendering and efficient MPM simulations, we achieve *real-time* performance for scenes with simple dynamics.

6.2.2 Related Work

Radiance Fields Rendering for View Synthesis. Radiance field methods have gained considerable interest in recent years due to their extraordinary ability to generate novel-view scenes and their great potential in 3D reconstruction. The adoption of deep learning techniques has led to the prominence of neural rendering and point-based rendering methods, both of which have inspired a multitude of subsequent works. On the one hand, the NeRF framework employs a fully-connected network to model one scene (Mildenhall et al., 2021). The network takes spatial position and viewing direction as inputs and produces the volume density and radiance color. These outputs are subsequently utilized in image generation through volume rendering techniques. Building upon the achievements of NeRF, further studies have focused on enhancing reconstruction quality and improving training speeds (Fridovich-Keil et al., 2022; Müller et al., 2022; Sun et al., 2022; Barron et al., 2022; Xu et al., 2022; Lin et al., 2022b). On the other hand, researchers have also investigated differentiable point-based methods for real-time rendering of unbounded scenes. Among the current investigations, the state-of-the-art results are achieved by the recently published 3D Gaussian Splatting framework (Kerbl et al., 2023). Contrary to prior implicit neural representations, GS employs an explicit and unstructured representation of one scene, offering the advantage of straightforward extension to post-manipulation. Moreover, its fast visibility-aware rendering

algorithm also enables real-world dynamics generations.

Dynamic Neural Radiance Field. An inherent evolution of the NeRF framework entails the integration of a temporal dimension to facilitate the representation of dynamic scenes. For example, both Pumarola et al. (2021b) and Park et al. (2021) decompose time-dependent neural fields into an inverse displacement field and canonical time-invariant neural fields. In this context, the trajectory of query rays is altered by the inverse displacement field and then positioned within the canonical space. Subsequent studies have adhered to the aforementioned design when exploring applications related to NeRF deformations, such as static scene editing and dynamic scene reconstruction (Li et al., 2023f; Peng et al., 2022; Yuan et al., 2022; Chen et al., 2022a; Qiao et al., 2022, 2023; Liu et al., 2023c). Additionally, Yuan et al. (2022); Qiao et al. (2022); Liu et al. (2023c) have contributed to the incorporation of physics-based deformations into the NeRF framework. However, the effectiveness of these methodologies relies on the usage of exported meshes derived from NeRFs. To circumvent this restriction, explicit geometric representations have been explored for forward displacement modeling (Xu et al., 2022; Kerbl et al., 2023). In particular, Chen et al. (2023a); Luiten et al. (2023); Yang et al. (2023a); Wu et al. (2023a); Yang et al. (2023b) directly manipulate NeRF fields. Li et al. (2023d) extends this approach by including physical simulators to achieve more dynamic behaviors. In this study, we leverage the explicit 3D Gaussian Splatting ellipsoids as a unified representation for both physics and graphics. In contrast to previous dynamic GS frameworks, which either maintain the shapes of Gaussian kernels or learn to modify them, our approach uniquely leverages the first-order information from the displacement map (deformation gradient) to assist dynamic simulations. In this way, we are able to deform the Gaussian kernels and seamlessly integrate the simulation within the GS framework.

Material Point Method. The Material Point Method (MPM) is a widely used simulation framework for a broad range of multi-physics phenomena (Hu et al., 2018a). The inherent capability of the MPM system allows for topology changes and frictional interactions, making it suitable for simulating various materials, including but not limited to elastic objects, fluids,

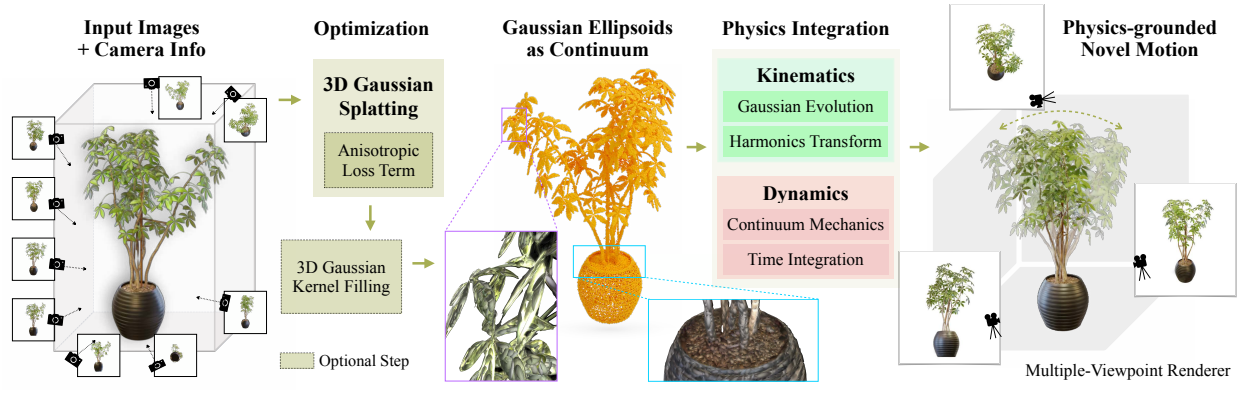


Figure 6.10: **Method Overview.** PhysGaussian is a unified simulation-rendering pipeline that incorporates 3D Gaussian splatting representation and continuum mechanics to generate physics-based dynamics and photo-realistic renderings simultaneously and seamlessly.

sand, and snow (Stomakhin et al., 2013; Jiang et al., 2015a; Klár et al., 2016). MPM can also be expanded to simulate objects that possess codimensional characteristics (Jiang et al., 2017a). In addition, the efficacy of utilizing GPU(s) to accelerate MPM implementations has also been demonstrated in (Gao et al., 2018b; Hu et al., 2019a; Wang et al., 2020b; Qiu et al., 2023). Owing to its well-documented advantages, we employ the MPM to support the latent physical dynamics. This choice allows us to efficiently import dynamics into various scenarios with a shared particle representation alongside the Gaussian Splatting framework.

6.2.3 Method Overview

We propose PhysGaussian (Figure 6.10), a unified simulation-rendering framework for generative dynamics based on continuum mechanics and 3D GS. Adopted from Kerbl et al. (2023), we first reconstruct a GS representation of a static scene, with an optional anisotropic loss term to regularize over-skinny kernels. These Gaussians are viewed as the discretization of the scene to be simulated. Under our novel kinematics, we directly splat the deformed Gaussians for photo-realistic renderings. For better physics compliance, we also optionally fill the internal regions of objects. We detail these in this section.

6.2.3.1 3D Gaussian Splatting

3D Gaussian Splatting method (Kerbl et al., 2023) reparameterizes NeRF (Mildenhall et al., 2021) using a set of unstructured 3D Gaussian kernels $\{\boldsymbol{x}_p, \sigma_p, \boldsymbol{A}_p, \mathcal{C}_p\}_{p\in\mathcal{P}}$, where $\boldsymbol{x}_p$, σ_p, $\boldsymbol{A}_p$, and $\mathcal{C}_p$ represent the centers, opacities, covariance matrices, and spherical harmonic coefficients of the Gaussians, respectively. To render a view, GS projects these 3D Gaussians onto the image plane as 2D Gaussians, differing from traditional NeRF techniques that emit rays from the camera. The final color of each pixel is computed as

$$\boldsymbol{C} = \sum_{k\in\mathcal{P}} \alpha_k \mathrm{SH}(\boldsymbol{d}_k; \mathcal{C}_k) \prod_{j=1}^{k-1}(1-\alpha_j). \tag{6.8}$$

Here α_k represents the z-depth ordered effective opacities, *i.e.*, products of the 2D Gaussian weights and their overall opacities σ_k; $\boldsymbol{d}_k$ stands for the view direction from the camera to $\boldsymbol{x}_k$. Per-view optimizations are performed using L_1 loss and SSIM loss. This explicit representation of the scene not only significantly accelerates training and rendering speeds, but also enables direct manipulation of the NeRF scene. The data-driven dynamics are supported by making $\boldsymbol{x}_p, A_p$ time-dependent (Wu et al., 2023a) and minimizing rendering losses over videos. In Section 6.2.3.1, we show that this time-dependent evolution can be given by the continuum deformation map.

6.2.3.2 Continuum Mechanics

Continuum mechanics describes motions by a time-dependent continuous deformation map $\boldsymbol{x} = \boldsymbol{\phi}(\boldsymbol{X}, t)$ between the undeformed material space Ω^0 and the deformed world space Ω^t at time t. The deformation gradient $F(\boldsymbol{X}, t) = \nabla_{\boldsymbol{X}}\boldsymbol{\phi}(\boldsymbol{X}, t)$ encodes local transformations including stretch, rotation, and shear (Bonet and Wood, 1997). The evolution of the deformation $\boldsymbol{\phi}$ is governed by the conservation of mass and momentum. Conservation of mass ensures that the mass within any infinitesimal region $B_\epsilon^0 \in \Omega^0$ remains constant over time:

$$\int_{B_\epsilon^t} \rho(\boldsymbol{x}, t) \equiv \int_{B_\epsilon^0} \rho(\boldsymbol{\phi}^{-1}(\boldsymbol{x}, t), 0), \tag{6.9}$$

where $B_\epsilon^t = \boldsymbol{\phi}(B_\epsilon^0, t)$ and $\rho(\boldsymbol{x}, t)$ is the density field characterizing material distribution. Denoting the velocity field with $\boldsymbol{v}(\boldsymbol{x}, t)$, the conservation of momentum is given by

$$\rho(\boldsymbol{x}, t)\dot{\boldsymbol{v}}(\boldsymbol{x}, t) = \nabla \cdot \boldsymbol{\sigma}(\boldsymbol{x}, t) + \boldsymbol{f}^{\text{ext}}, \tag{6.10}$$

where $\boldsymbol{\sigma} = \frac{1}{\det(\boldsymbol{F})}\frac{\partial \Psi}{\partial \boldsymbol{F}}(\boldsymbol{F}^E)\boldsymbol{F}^{E^T}$ is the Cauchy stress tensor associated with a hyperelastic energy density $\Psi(\boldsymbol{F})$, and $\boldsymbol{f}^{\text{ext}}$ is the external force per unit volume (Bonet and Wood, 1997; Jiang et al., 2016). Here the total deformation gradient can be decomposed into an elastic part and a plastic part $\boldsymbol{F} = \boldsymbol{F}^E\boldsymbol{F}^P$ to support permanent rest shape changes caused by plasticity. The evolution of $\boldsymbol{F}^E$ follows some specific plastic flow such that it is always constrained within a predefined elastic region (Bonet and Wood, 1997).

6.2.3.3 Material Point Method

Material Point Method (MPM) solves the above governing equations by combining the strengths of both Lagrangian particles and Eulerian grids (Stomakhin et al., 2013; Jiang et al., 2016). The continuum is discretized by a collection of particles, each representing a small material region. These particles track several time-varying Lagrangian quantities such as position $\boldsymbol{x}_p$, velocity $\boldsymbol{v}_p$, and deformation gradient $\boldsymbol{F}_p$. The mass conservation in Lagrangian particles ensures the constancy of total mass during movement. Conversely, momentum conservation is more natural in Eulerian representation, which avoids mesh construction. We follow Stomakhin et al. (2013) to integrate these representations using C^1 continuous B-spline kernels for two-way transfer. From time step t^n to t^{n+1}, the momentum conservation, discretized by the forward Euler scheme, is represented as

$$\frac{m_i}{\Delta t}(\boldsymbol{v}_i^{n+1} - \boldsymbol{v}_i^n) = -\sum_p V_p^0 \frac{\partial \Psi}{\partial \boldsymbol{F}}(\boldsymbol{F}_p^{E,n})\boldsymbol{F}_p^{E,n^T}\nabla w_{ip}^n + \boldsymbol{f}_i^{ext}. \tag{6.11}$$

Here i and p represent the fields on the Eulerian grid and the Lagrangian particles respectively; w_{ip}^n is the B-spline kernel defined on i-th grid evaluated at $\boldsymbol{x}_p^n$; V_p^0 is the initial representing volume, and Δt is the time step size. The updated grid velocity field $\boldsymbol{v}_i^{n+1}$ is transferred

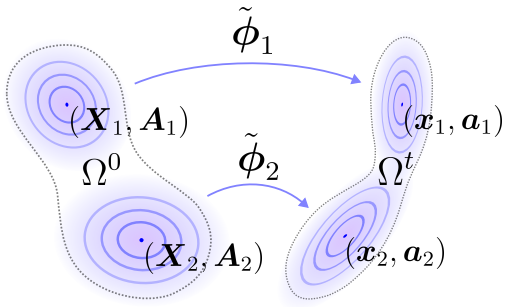


Figure 6.11: Illustration of kernel deformation.

back onto particle to v_p^{n+1}, updating the particles' positions to $\boldsymbol{x}_p^{n+1} = \boldsymbol{x}_p^n + \Delta t \boldsymbol{v}_p^{n+1}$. We track $\boldsymbol{F}^E$ rather than both $\boldsymbol{F}$ and $\boldsymbol{F}^P$ (Simo and Hughes, 1998), which is updated by $\boldsymbol{F}_p^{E,n+1} = (\boldsymbol{I} + \Delta t \nabla \boldsymbol{v}_p)\boldsymbol{F}_p^{E,n} = (\boldsymbol{I} + \Delta t \sum_i \boldsymbol{v}_i^{n+1} \nabla w_{ip}^{n\,T})\boldsymbol{F}_p^{E,n}$ and regularized by an additional return mapping to support plasticity evolution: $\boldsymbol{F}_p^{E,n+1} \leftarrow \mathcal{Z}(\boldsymbol{F}_p^{E,n+1})$. Different plasticity models define different return mappings. We refer to the supplemental document for details of the simulation algorithm and different return mappings.

6.2.3.4 Physics-Integrated 3D Gaussians

We treat Gaussian kernels as discrete particle clouds for spatially discretizing the simulated continuum. As the continuum deforms, we let the Gaussian kernels deform as well. However, for a Gaussian kernel defined at $\boldsymbol{X}_p$ in the material space, $G_p(\boldsymbol{X}) = e^{-\frac{1}{2}(\boldsymbol{X}-\boldsymbol{X}_p)^T \boldsymbol{A}_p^{-1}(\boldsymbol{X}-\boldsymbol{X}_p)}$, the deformed kernel under the deformation map $\boldsymbol{\phi}(\boldsymbol{X}, t)$,

$$G_p(\boldsymbol{x}, t) = e^{-\frac{1}{2}(\boldsymbol{\phi}^{-1}(\boldsymbol{x},t)-\boldsymbol{X}_p)^T \boldsymbol{A}_p^{-1}(\boldsymbol{\phi}^{-1}(\boldsymbol{x},t)-\boldsymbol{X}_p)} \tag{6.12}$$

is not necessarily Gaussian in the world space, which violates the requirements of the splatting process. Fortunately, if we assume particles undergo local affine transformations characterized by the first-order approximation

$$\tilde{\phi}_p(\boldsymbol{X}, t) = \boldsymbol{x}_p + \boldsymbol{F}_p(\boldsymbol{X} - \boldsymbol{X}_p), \tag{6.13}$$

the deformed kernel becomes Gaussian as desired:

$$G_p(\boldsymbol{x}, t) = e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}_p)^T(\boldsymbol{F}_p\boldsymbol{A}_p\boldsymbol{F}_p^T)^{-1}(\boldsymbol{x}-\boldsymbol{x}_p)}. \tag{6.14}$$

This transformation naturally provides a time-dependent version of $\boldsymbol{x}_p$ and $\boldsymbol{A}_p$ for the 3D GS framework (Figure 6.11):

$$\begin{aligned} \boldsymbol{x}_p(t) &= \boldsymbol{\phi}(\boldsymbol{X}_p, t), \\ \boldsymbol{a}_p(t) &= \boldsymbol{F}_p(t)\boldsymbol{A}_p\boldsymbol{F}_p(t)^T. \end{aligned} \tag{6.15}$$

In summary, given the 3D GS of a static scene $\{\boldsymbol{X}_p, \boldsymbol{A}_p, \sigma_p, \mathcal{C}_p\}$, we use simulation to dynamize the scene by evolving these Gaussians to produce dynamic Gaussians $\{\boldsymbol{x}_p(t), \boldsymbol{a}_p(t), \sigma_p, \mathcal{C}_p\}$. Here we assume that the opacity and the coefficients of spherical harmonics are invariant over time, but the harmonics will be rotated as discussed in the next section. We also initialize other physical quantities in Equation (6.11): the representing volume of each particle V_p^0 is initialized as background cell volume divided by the number of contained particles; the mass m_p is then inferred from user-specified density ρ_p as $m_p = \rho_p V_p^0$. To render these deformed Gaussian kernels, we use the splatting from the original GS framework (Kerbl et al., 2023). It should be highlighted that the integration of physics into 3D Gaussians is **seamless**: on the one hand, the Gaussians themselves are viewed as the discretization of the continuum, which can be simulated directly; on the other hand, the deformed Gaussians can be directly rendered by the splatting procedure, avoiding the need for commercial rendering software in traditional animation pipelines. Most importantly, we can directly simulate scenes reconstructed from real data, achieving WS^2.

6.2.3.5 Evolving Orientations of Spherical Harmonics

Rendering the world-space 3D Gaussians can already obtain high-quality results. However, when the object undergoes rotations, the spherical harmonic bases are still represented in the material space, resulting in varying appearances even if the view direction is fixed relatively to the object. The solution is simple: when an ellipsoid is rotated over time, we rotate the

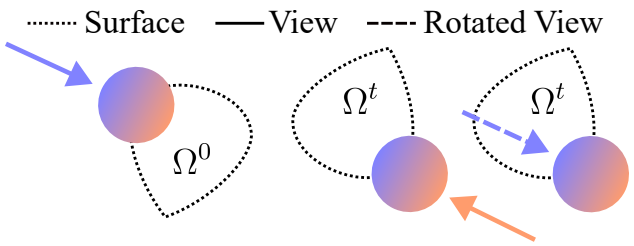


Figure 6.12: Rotations of spherical harmonics.

orientations of its spherical harmonics as well. However, the bases are hard-coded inside the GS framework. We equivalently achieve this evolution by applying inverse rotation on view directions. This effect is illustrated in Figure 6.12. We remark that the rotation of view directions is not considered in Wu et al. (2023a). Chen et al. (2023a) tackles this issue in the Point-NeRF framework, but requires tracking of surface orientation. In our framework, the local rotation is readily obtained in the deformation gradient $\boldsymbol{F}_p$. Denote $f^0(\boldsymbol{d})$ as a spherical harmonic basis in material space, with $\boldsymbol{d}$ being a point on the unit sphere (indicating view direction). The polar decomposition, $\boldsymbol{F}_p = \boldsymbol{R}_p\boldsymbol{S}_p$, leads us to the rotated harmonic basis:

$$f^t(\boldsymbol{d}) = f^0(\boldsymbol{R}^T\boldsymbol{d}). \tag{6.16}$$

6.2.3.6 Incremental Evolution of Gaussians

We also propose an alternative way for Gaussian kinematics that better fits the updated Lagrangian framework, which avoids the dependency on the total deformation gradient $\boldsymbol{F}$. This approach also paves the way for physical material models that do not rely on employing $\boldsymbol{F}$ as the strain measure. Following conventions from computational fluid dynamics (McKIVER and Dritschel, 2003; Chandrasekhar, 1967), the update rule for the world-space covariance matrix $\boldsymbol{a}$ can also be derived by discretizing the rate form of kinematics $\dot{\boldsymbol{a}} = (\nabla\boldsymbol{v})\boldsymbol{a} + \boldsymbol{a}(\nabla\boldsymbol{v})^T$:

$$\boldsymbol{a}_p^{n+1} = \boldsymbol{a}_i^n + \Delta t(\nabla\boldsymbol{v}_p\boldsymbol{a}_p^n + \boldsymbol{a}_p^n\nabla\boldsymbol{v}_p^T). \tag{6.17}$$

This formulation facilitates the incremental update of the Gaussian kernel shapes from time step t^n to t^{n+1} without the need to obtain $\boldsymbol{F}_p$. The rotation matrix $\boldsymbol{R}_p$ of each spherical

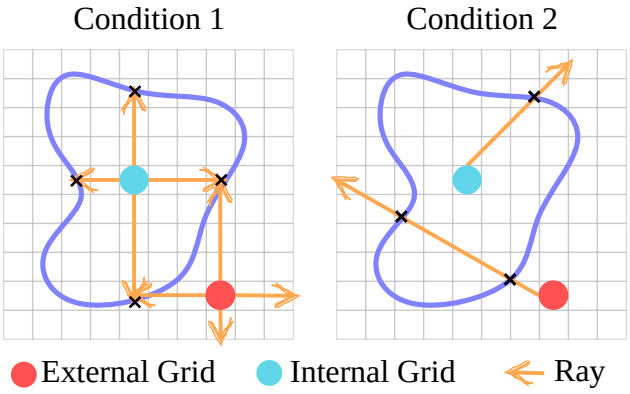


Figure 6.13: Conditions for internal filling.

harmonics basis can be incrementally updated in a similar manner. Starting from $\boldsymbol{R}_p^0 = \boldsymbol{I}$, we extract the rotation matrix $\boldsymbol{R}_p^{n+1}$ from $(\boldsymbol{I} + \Delta t \boldsymbol{v}_p)\boldsymbol{R}_p^n$ using the polar decomposition.

6.2.3.7 Internal Filling

The internal structure is occluded by the object's surface, as the reconstructed Gaussians tend to distribute near the surface, resulting in inaccurate behaviors of volumetric objects. To fill particles into the void internal region, inspired by Tang et al. (2023), we borrow the 3D opacity field from 3D Gaussians

$$d(\boldsymbol{x}) = \sum_p \sigma_p \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_p)^T \boldsymbol{A}_p^{-1}(\boldsymbol{x} - \boldsymbol{x}_p)\right). \tag{6.18}$$

This continuous field is discretized onto a 3D grid. To achieve robust internal filling, we first define the concept of "intersection" within the opacity field, guided by a user-defined threshold σ_{th}. Specifically, we consider it an intersection when a ray passes from a lower opacity grid ($\sigma_i < \sigma_{th}$) to a higher opacity one ($\sigma_j > \sigma_{th}$). Based on this definition, we identify candidate grids by casting rays along 6 axes and checking intersections (condition 1). Rays originating from internal cells will always intersect with the surface. To further refine our selection of candidate grids, we employ an additional ray to assess the intersection number (condition 2), thus ensuring greater accuracy. These two conditions are illustrated in Figure 6.13.

Visualization of these internal particles is also crucial as they may get exposed due to large deformation. Those filled particles inherit $\sigma_p, \mathcal{C}_p$ from their closet Gaussian kernels. Each particle's covariance matrix is initialized as $\text{diag}(r_p^2, r_p^2, r_p^2)$, where r is the particle radius calculated from its volume: $r_p = (3V_p^0/4\pi)^{\frac{1}{3}}$. Alternatively, one may also consider employing generative models for internal filling, potentially leading to more realistic results.

6.2.3.8 Anisotropy Regularizer

The anisotropy of Gaussian kernels increases the efficiency of 3D representation while over-skinny kernels may point outward from the object surface under large deformations, leading to unexpected plush artifacts. We propose the following training loss during 3D Gaussian reconstruction:

$$\mathcal{L}_{aniso} = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \max\{\max(\boldsymbol{S}_p)/\min(\boldsymbol{S}_p), r\} - r, \tag{6.19}$$

where $\boldsymbol{S}_p$ are the scalings of 3D Gaussians (Kerbl et al., 2023). This loss essentially constrains that the ratio between the major axis length and minor axis length does not exceed r. If desired, this term can be added to the training loss.

6.2.4 Experiments

In this section, we show the versatility of our approach across a wide range of materials. We also evaluate the effectiveness of our method across a comprehensive suite of benchmarks.

6.2.4.1 Evaluation of Generative Dynamics

Datasets We evaluate our method for generating diverse dynamics using several sources of input. In addition to the synthetic data (*sofa suite*) generated by BlenderNeRF (Raafat, 2023), we utilize *fox*, *plane*, and *ruins* from the datasets of Instant-NGP (Müller et al., 2022), Nerfstudio (Tancik et al., 2023) and the DroneDeploy NeRF (Pilkington, 2022), respectively. Furthermore, we collect two real-world datasets (referred to as *toast* and *jam*) with an iPhone. Each scene contains 150 photos. The initial point clouds and camera parameters are obtained

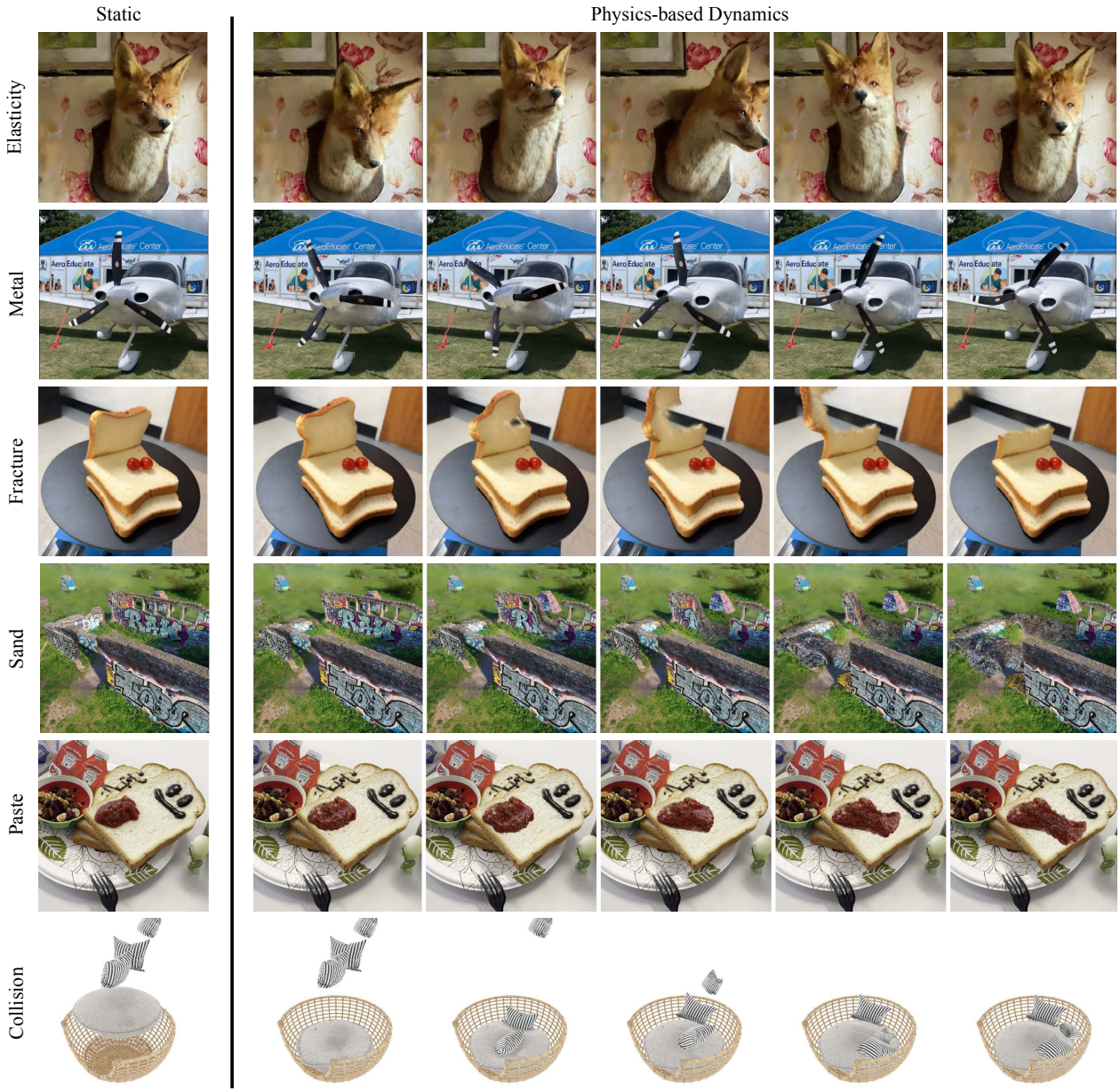


Figure 6.14: **Material Versatility.** We demonstrate exceptional versatility of our approach across a wide variety of examples: *fox* (elastic entity), *plane* (plastic metal), *toast* (fracture), *ruins* (granular material), *jam* (viscoplastic material), and *sofa suite* (collision).

using COLMAP (Schönberger et al., 2016; Schönberger and Frahm, 2016).

Simulation Setups We build upon the MPM from Zong et al. (2023). To generate novel physics-based dynamics of a 3D Gaussian scene, we manually select a simulation region and normalize it to a cube with edge length 2. The internal particle filling can be performed

before simulation. The cuboid simulation domain is discretized by a 3D dense grid. We selectively modify the velocities of specific particles to induce controlled movement. The remaining particles follow natural motion patterns governed by the established physical laws. All our experiments are performed on a 24-core 3.50GHz Intel i9-10920X machine with a Nvidia RTX 3090 GPU.

Results We simulate a wide range of physics-based dynamics. For each type of dynamics, we visualize one example with its initial scene and deformation sequence, as shown in Figure 6.14. Additional experiments are included in the supplemental document. The dynamics include: **Elasticity** refers to the property where the rest shape of the object remains invariant during deformation, representing the simplest form of daily-life dynamics. **Metal** can undergo permanent rest shape changes, which follows von-Mises plasticity model. **Fracture** is naturally supported by MPM simulation, where large deformations can cause particles to separate into multiple groups. **Sand** follows Druker-Prager plasticity model (Klár et al., 2016), which can capture granular-level frictional effects among particles. **Paste** is modeled as viscoplastic non-Newtonian fluid, adhering to Herschel-Bulkley plasticity model (Yue et al., 2015). **Collision** is another key feature of MPM simulation, which is automatically handled by grid time integration. Explicit MPM can be highly optimized to run on GPUs. We highlight that some of the cases can achieve real-time based on the 1/24-s frame duration: *plane* (30 FPS), *toast* (25 FPS) and *jam* (36 FPS). While utilizing FEM may further accelerate the elasticity simulation, it will involve an additional step of mesh extraction and lose the generalizability of MPM in inelasticity simulation.

6.2.4.2 Lattice Deformation Benchmarks

Dataset Due to the absence of ground truth for post-deformation, we utilize BlenderNeRF (Raafat, 2023) to synthesize several scenes, applying bending and twisting with the lattice deformation tool. For each scene, we create 100 multi-view renderings of the undeformed state for training, and 100 multi-view renderings of each deformed state to serve as ground truth for the deformed NeRFs. The lattice deformations are set as input to all methods for

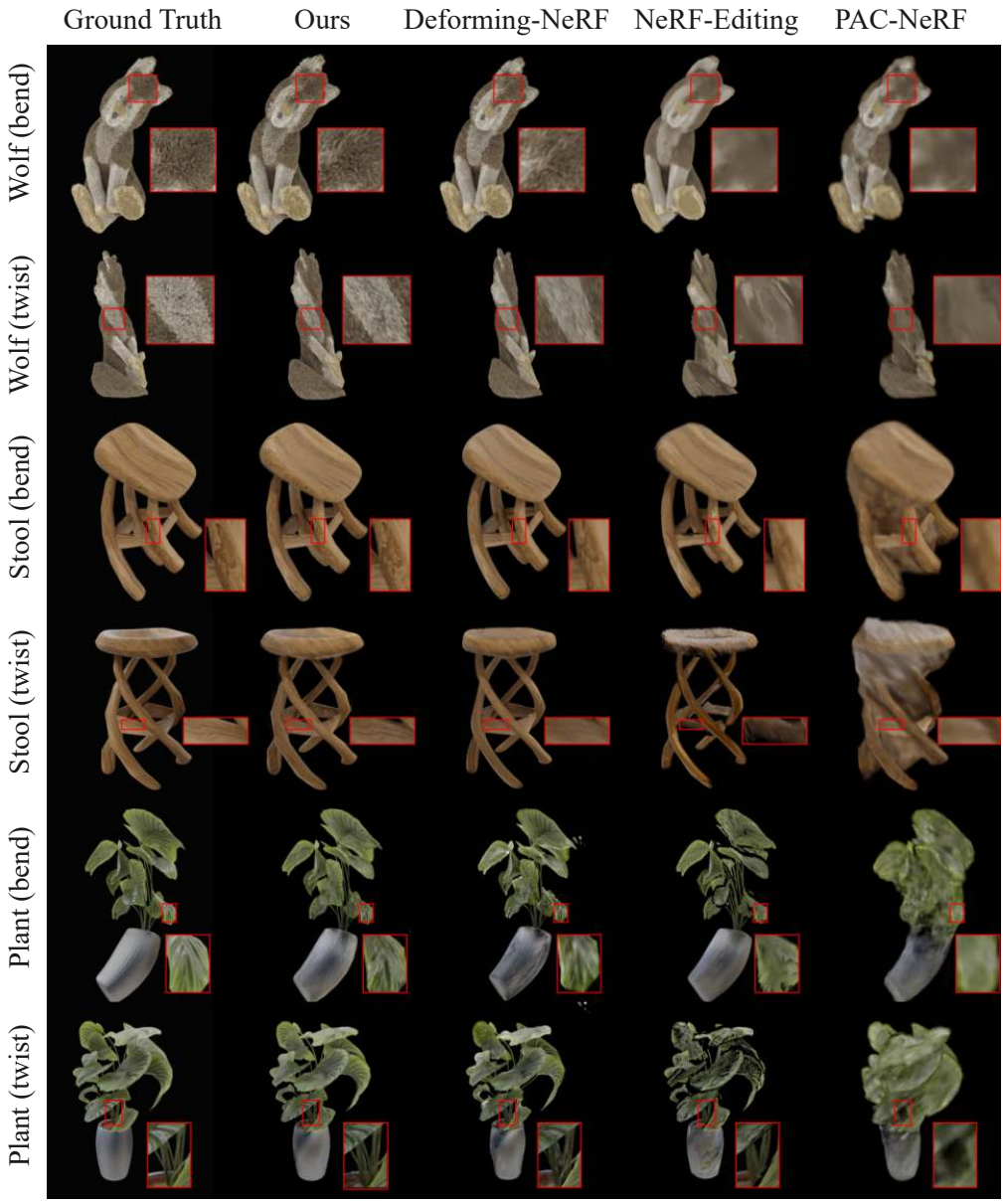


Figure 6.15: **Comparisons.** For each benchmark case, we select one test viewpoint and visualize all comparisons. We zoom in on some regions to highlight the ability of our method to maintain high-fidelity rendering quality after deformations. We use a black background to avoid the interference of the background.

fair comparisons.

Comparisons We compare our method with several state-of-the-art NeRF frameworks that support manual deformations: **1) NeRF-Editing (Yuan et al., 2022)** deforms NeRF using an extracted surface mesh, **2) Deforming-NeRF (Xu and Harada, 2022)** utilizes a

Figure 6.16: **Ablation Studies.** Non-extensible Gaussians can lead to severe visual artifacts during deformations. Although direct rendering the deformed Gaussian kernels can already obtain good results, additional rotations on spherical harmonics can improve the rendering quality.

cage mesh for deformation, and **3) PAC-NeRF (Li et al., 2023d)** manipulates individual initial particles.

We show qualitative results in Figure 6.15 and quantitative results in Table 6.4. NeRF-Editing uses NeuS (Wang et al., 2021) as the scene representation, which is more suited for surface reconstructions rather than high-fidelity renderings. Consequently, its rendering quality is inherently lower than that of 3DGS. Furthermore, the deformation highly depends on the precision of the extracted surface mesh and the dilated cage mesh – an overly tight mesh might not encompass the entire radiance field, while an excessively large one could result in a void border, as observed in the twisting stool and plant examples. Deforming-NeRF, on the other hand, provides clear renderings and potentially leads to enhanced results if higher-resolution deformation cages are provided. However, it employs a smooth interpolation from all cage vertices, thus filtering out fine local details and failing to match lattice deformations. PAC-NeRF is designed for simpler objects and textures in system identification tasks. While offering flexibility through its particle representation, it does not achieve high rendering fidelity. Our method utilizes both zero-order information (the deformation map) and first-order information (the deformation gradient) from each lattice cell. It outperforms the other methods across all cases, as high rendering qualities are well preserved after deformations. Although not primarily designed for editing tasks, this comparison showcases our method's significant potential for realistic editing of static NeRF scenes.

Ablation Studies We further conduct several ablation studies on these benchmark scenes to validate the necessity of the kinematics of Gaussian kernels and spherical harmonics:

Table 6.4: We synthesize a lattice deformation benchmark dataset to compare with baselines and conduct ablation studies to validate our design choices. PSNR scores are reported (higher is better). Our method outperforms the others across all cases.

Test Case	Wolf		Stool		Plant	
Deformation Operator	Bend	Twist	Bend	Twist	Bend	Twist
NeRF-Editing (Yuan et al., 2022)	26.74	24.37	25.00	21.10	19.85	19.08
Deforming-NeRF (Xu and Harada, 2022)	21.65	21.72	22.32	21.16	17.90	18.63
PAC-NeRF (Li et al., 2023d)	26.91	25.27	21.83	21.26	18.50	17.78
Fixed Covariance	26.77	26.02	29.94	25.31	23.95	23.09
Rigid Covariance	26.84	26.16	30.28	25.70	24.09	23.53
Fixed Harmonics	26.83	26.02	30.87	25.75	25.09	23.69
Ours	**26.96**	**26.46**	**31.15**	**26.15**	**25.81**	**23.87**

1) Fixed Covariance only translates the Gaussian kernels. **2) Rigid Covariance** only applies rigid transformations on the Gaussians, as assumed in Luiten et al. (2023). **3) Fixed Harmonics** does not rotate the orientations of spherical harmonics, as assumed in Wu et al. (2023a).

Here we visualize one example in Figure 6.16. We can observe that Gaussians will not properly cover the surface after deformation if they are non-extensible, leading to severe visual artifacts. Enabling the rotation of spherical harmonics can slightly improve the consistency with the ground truth. We include quantitative results on all test cases in Table 6.4, which shows that all these enhancements are needed to achieve the best performance of our method.

6.2.4.3 Additional Qualitative Studies

Internal Filling Typically, the 3D Gaussian splatting framework focuses on the surface appearance of objects and often fails to capture their internal structure. Consequently, the interior of the modeled object remains hollow, resembling a mere shell. This is usually not true in the real world, leading to unrealistic simulation results. To address this challenge, we introduce an internal filling method leveraging a reconstructed density field, which is derived from the opacity of Gaussian kernels. Figure 6.17 showcases our simulation results with varying physical parameters. Objects devoid of internal particles tend to collapse when subjected to gravity forces, irrespective of the material parameters used. In contrast,

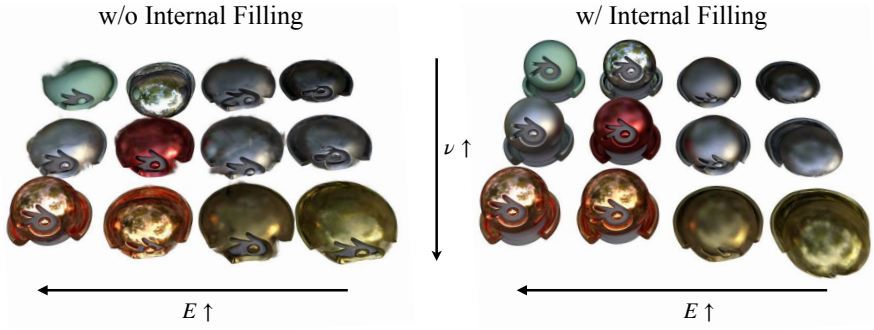


Figure 6.17: **Internal filling** enables more realistic simulation results. Our method also supports flexible control of dynamics via material parameters. A larger Young's modulus E indicates higher stiffness while a larger poission ratio ν leads to better volume preservation.

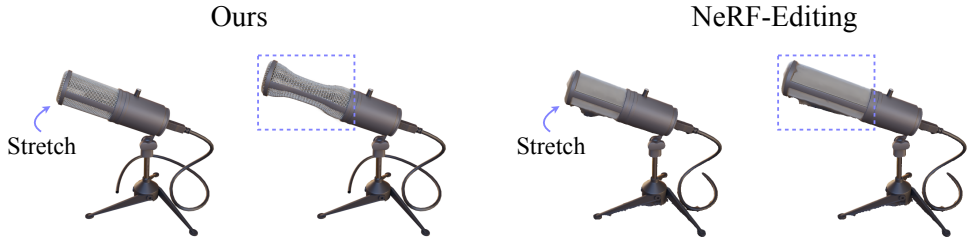


Figure 6.18: **Volume Conservation.** Compared to the geometry-based editing method (Yuan et al., 2022), our physics-based method is able to capture volumetric behaviors, leading to more realistic dynamics.

our approach assisted by internal filling allows for nuanced control over object dynamics, effectively adjusting to different material characteristics.

Volume Conservation Existing approaches to NeRF manipulation focus primarily on geometric adjustments without incorporating physical properties. A key attribute of real-world objects is their inherent ability to conserve volume during deformation. In Figure 6.18, we conduct a comparison study between our method and NeRF-Editing (Yuan et al., 2022), which utilizes surface As-Rigid-As-Possible (ARAP) deformation (Sorkine and Alexa, 2007). Unlike NeRF-Editing, our approach accurately captures and maintains the volume of the deformed objects.

Anisotropy Regularizer 3D Gaussian models inherently represent anisotropic ellipsoids. However, excessively slender Gaussian kernels can lead to burr-like visual artifacts, especially

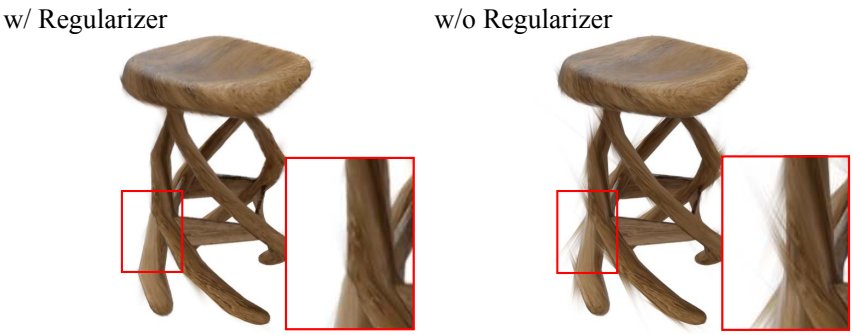


Figure 6.19: **Anisotropy Regularizer.** We introduce an anisotropy constraint for Gaussian kernels, effectively enhancing the fidelity of the Gaussian-based representation under dynamic conditions.

pronounced during large deformations To tackle this issue, we introduce an additional regularization loss Equation (6.19) to constrain anisotropy. As demonstrated in Figure 6.19, this additional loss function effectively mitigates the artifacts induced by extreme anisotropy.

Additional Evaluations We present additional evaluations of our method in Figure 6.20. The *vasedeck* data is from the Nerf dataset (Mildenhall et al., 2021) and the others are synthetic data, generated using BlenderNeRF (Raafat, 2023).

6.2.5 Discussion

Conclusion This paper introduces PhysGaussian, a unified simulation-rendering pipeline that generates physics-based dynamics and photo-realistic renderings simultaneously and seamlessly.

Limitation In our framework, the evolution of shadows is not considered, and material parameters are manually set. Automatic parameter assignment could be derived from videos by combining GS segmentation (Cen et al., 2023; Ye et al., 2023) with a differentiable MPM simulator. Additionally, incorporating geometry-aware 3DGS reconstruction methods (Guédon and Lepetit, 2023) could enhance generative dynamics. Future work will also explore handling more versatile materials like liquids and integrating more intuitive user controls, possibly leveraging advancements in Large Language Models (LLMs).

Figure 6.20: **Additional Evaluation.** Examples from top to bottom are: *vasedeck* (elastic entity), *bread* (fracture), *cake* (viscoplastic material), *can* (plastic metal) and *wolf* (granular material).

6.3 VR-GS: A Physical Dynamics-Aware Interactive Gaussian Splatting System in Virtual Reality

6.3.1 Introduction

As digital technology advances, the importance of 3D content is becoming increasingly prominent across various industries, from entertainment to education. This growth is driving the demand for high-fidelity 3D content within the Computer Graphics (CG) and Computer Vision (CV) communities. Despite their capabilities of being rendered efficiently, traditional

3D/4D content creation, reliant on 3D modeling tools and game engines, is time-intensive and complex, often beyond the reach of non-expert users. This accessibility barrier limits broader user engagement in high-quality content creation.

Recognizing the advantages and limitations of the traditional graphics pipeline, our goal is to find a modern variant. To enhance the visual quality and ease of creation for non-expert users, we move away from traditional 3D models in graphics pipelines and instead, adopt state-of-the-art radiance field techniques for rendering. In this context, Neural Radiance Fields (NeRF) emerge as a natural choice. Despite its versatility, NeRF's volume rendering falls short in efficiency for interactive applications, which demand high frame rates. Additionally, NeRF's approach to handling deformations—requiring bending of query rays via an inverse deformation map—is slow (Pumarola et al., 2021b). Fortunately, 3D Gaussian Splatting (GS) (Kerbl et al., 2023) has been recently introduced as an efficient and explicit alternative to NeRF. This method not only excels in rendering efficiency but also provides an explicit geometric representation that can be directly deformed or edited. The explicit nature of 3D GS can simplify the direct manipulation of geometry by solving a Partial Differentiable Equation (PDE) that governs the motions of Gaussian kernels. Furthermore, GS eliminates the need for high-fidelity meshes, UV maps, and textures, offering the potential for naturally photorealistic appearances. It is worth noting that many studies have already demonstrated the use of 3D GS for 4D dynamics (Park et al., 2021; Pumarola et al., 2021b; Yang et al., 2023b) and the integration of physics-based simulations for more realistic 4D content generation (Xie et al., 2023b; Feng et al., 2023), as well as for creating animatable avatars (Zielonka et al., 2023; Xu et al., 2023c).

In this paper, we introduce a physics-aware interactive system for immersive manipulation of 3D content represented with GS. To ensure an interactive experience, we utilize Position-based Dynamics (PBD) (Macklin et al., 2016), a highly adaptable and unified physical simulator, for real-time deformation simulation. Direct incorporation of a simulator onto GS kernels presents challenges, as the simulation and rendering processes have distinct geometrical representations. To address this, we construct a tetrahedral cage for each segmented GS kernel group and embed these kernel groups into corresponding meshes. The deformed

Figure 6.21: **Animal Crossing.** Utilizing our system, individuals can engage in intuitive and interactive physics-based game-play with deformable virtual animals and realistic environments represented with 3D Gaussian Splatting.

mesh, driven by PBD, subsequently guides the deformation of the GS kernels. Noticing that simplistic embedding techniques can lead to undesirable, spiky deformations in GS kernels, we propose a novel two-level embedding approach. This method allows each Gaussian kernel to adapt to a smoothed average deformation of the surrounding tetrahedra. The intricate combination of GS and PBD through our two-level embedding not only achieves real-time physics-based dynamics but also upholds high-quality, realistic rendering. In summary, our contributions include:

- *A Physics-Aware Interactive System:* Development of a system that enables interactive, physics-aware manipulation of 3D content represented with GS.
- *Two-Level Deformation Embedding:* Introduction of a novel two-level embedding approach that allows Gaussians to adapt smoothly to the mesh, enhancing the deformation realism and preventing undesirable spiky artifacts.

We deploy our system on VR devices, enriched by segmentation, inpainting, and shadow map, offering users a rich platform for 3D content manipulation.

6.3.2 Related Work

Radiance Fields Rendering A variety of 3D representations, such as mesh (Sorkine and Cohen-Or, 2004), point clouds (Qi et al., 2017), signed distance fields (Wang et al., 2021), and grids (Zhu and Bridson, 2005) are exploited in early work for visual computing tasks,

including synthesis, estimation, manipulation, animation, reconstruction, and transmission of data about objects and scenes (Kerbl et al., 2023; Xie et al., 2022b; Gao et al., 2023). Since neural radiance fields (NeRF) (Mildenhall et al., 2021) was proposed for novel view synthesis with differentiable volume rendering, it has been applied to versatile computer graphics and computer vision tasks, such as scene reconstruction (Oechsle et al., 2021; Meng et al., 2023; Yariv et al., 2021), synthesis (Sun et al., 2023; Huang et al., 2023a), rendering (Lombardi et al., 2021; Wu et al., 2023b), interactive games (Xia et al.) and animation (Chen et al., 2021a; Peng et al., 2021), simulation (Li et al., 2023e), etc. While Neural Radiance Fields (NeRF) have achieved remarkable image quality, they suffer from significant time and memory inefficiencies (Lindell et al., 2021). To mitigate these issues, various methods have been introduced. Sparse representations (Liu et al., 2020), decomposed strategies (Niemeyer and Geiger, 2021; Rebain et al., 2021), and multi-resolution encoding (Müller et al., 2022) have all been proposed to enhance volume rendering efficiency while maintaining high quality. However, NeRFs are implicit representations, making it challenging to detect and resolve collisions (Qiao et al., 2023). Recently, 3D Gaussian splatting (GS) proposed by (Kerbl et al., 2023) utilizes an array of 3D Gaussian kernels to represent scenes explicitly, facilitating faster optimization and achieving state-of-the-art results.

Editing in Radiance-based Scene A natural extension of NeRF is to support user-guided editing. Li and Pan (2023) made use of two proxy cages to offer interactive control of the shape deformation of NeRF. Besides geometry editing, text prompts (Wang et al., 2023a) or a user-edited image (Bao et al., 2023) can also be adopted to conduct style transfer of NeRF. Besides, Lin et al. (2023a) put forward exploiting 2D sketches to edit high-quality facial NeRF. Concerning NeRF texture editing, Huang et al. (2023a) utilized a coarse-fine disentanglement representation and a patch-matching algorithm to synthesize textures of different shapes from multi-view images. De-Nerf (Wu et al., 2023b) exploited a hybrid light representation that enables users to relight NeRF. In contrast, Gaussian Splatting is more appropriate for post-editing tasks thanks to its explicit nature and has inspired a series of follow-up works (Chen et al., 2023b; Fang et al., 2023; Ye et al., 2023; Duisterhof et al., 2023;

Huang et al., 2023b).

Another significant direction in NeRF research is the incorporation of dynamic components into static scenes. To achieve this, a popular strategy (Pumarola et al., 2021b; Park et al., 2021) is to consider time as an additional input to the system, This method typically splits a dynamic NeRF into a canonical static field and a deformation field, with the latter mapping this canonical representation to a deformed one. More recently, Yang et al. (2023b) drew inspiration from 3D GS to reconstruct a dynamic scene using 4D Gaussian primitives that change over time. However, the dynamics in these methods are generally confined to motions captured from input data, limiting their ability to synthesize unseen dynamics. To generate novel dynamics, Xie et al. (2023b) and Feng et al. (2023) have integrated physics-based simulations into 3D static representations, paving the way for generating physically plausible and novel dynamic scenes.

Real-time Neural Radiance Fields Rendering NeRF is computationally expensive for real-time applications, such as VR, which causes high latency and low quality (Song et al., 2023; Li et al., 2022b). To address these challenges and enable high-fidelity rendering with minimal latency on a single GPU, various techniques have been proposed. These include gaze-contingent 3D neural representations (Deng et al., 2022), variable rate shading (Rolff et al., 2023), and hybrid surface-volume representations (Turki et al., 2023), all aimed at accelerating NeRF. In contrast to single-GPU solutions, VR-NeRF (Xu et al., 2023b) leveraged multiple GPUs to achieve high-quality volumetric rendering of NeRF. Beyond software acceleration techniques, RT-NeRF (Li et al., 2022a) utilized an algorithm-hardware co-design framework to provide real-time NeRF solutions for immersive VR rendering. While these methods primarily focus on enhancing NeRF rendering, our work is dedicated to facilitating interactive editing.

Li et al. (2023b) integrated affine transformation with NeRF to enable interactive features, such as exocentric manipulation, editing, and VR tunneling effects. However, their approach is limited to filter-based edits such as edge blurring and fails to support the deformation of virtual objects' geometry. RealityGit (Li et al., 2023a) and Magic NeRF Lens (Li et al.,

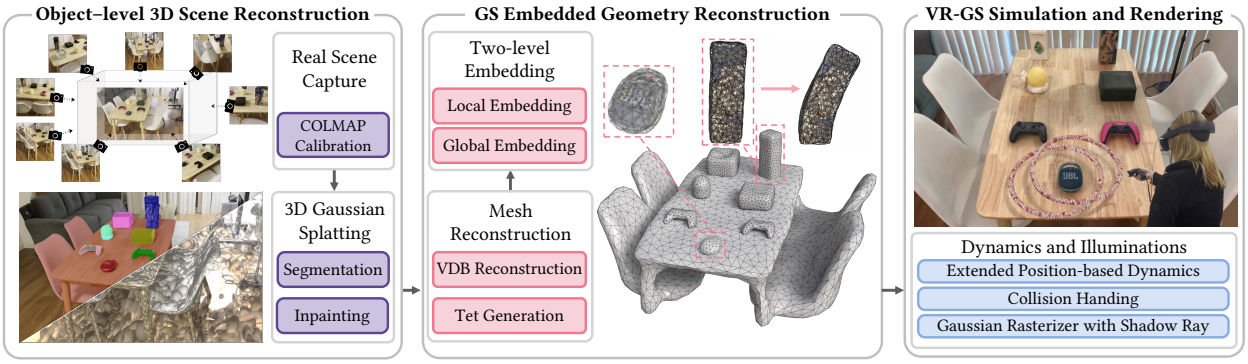


Figure 6.22: VR-GS is an interactive system designed to integrate 3D Gaussian Splatting (GS) and Position-based Dynamics (PBD) for generating a real-time interactive experience. Beginning with multi-view images, the pipeline combines scene reconstruction, segmentation, and inpainting using Gaussian kernels. These kernels form the foundation for VR-GS's utilization of the sparse volumetric data structure VDB, facilitating bounding mesh reconstruction and subsequent tetrahedralization. VR-GS further harnesses a novel two-level Gaussian embedding, PBD, collision detection, and shadow casting techniques, all converging to deliver a captivating and immersive user experience.

2023c) modified a NeRF model by erasing or revealing a portion of NeRF. Nevertheless, neither system permits deforming the geometrical structure of the model. In contrast to these works which concentrate on interactive editing of NeRF, our system enables users to interact with 3D GS in a physically realistic way in real time.

6.3.3 System Design

We propose a unified physics dynamics-aware interactive Virtual Reality (VR) system for real-time interactions with Gaussian Splatting (GS).

Immersive and Realistic Generative Dynamics Our targeted system aims to offer users an immersive and realistic experience, characterized by dynamic motions, 3D virtual shapes, and illuminations that closely mirror the real world (Kalawsky, 1999). This goal sets our system apart, especially in how it handles physics-based 3D deformations and dynamics, as opposed to those that rely solely on geometric deformation. The proposed system leverages physics-based simulation techniques to produce realistic dynamics and facilitate 3D shape editing. Unlike methods that reconstruct motion from time-dependent datasets or utilize

generative AI to drive Gaussian models, our system adheres to physical principles. To create an immersive experience, we choose VR as the interaction platform.

Real-Time Interaction For an interactive system, low latency is crucial to prevent disorientation or motion sickness (Sutcliffe et al., 2019; Deng et al., 2022). To offer instantaneous output and feedback, most interactive systems support basic transformations including rotation, scaling, and translation. Our system additionally offers users real-time physics-based interaction. To our knowledge, VR-GS represents the first interactive physics-based GS editor. To achieve both real-time performance and physically realistic editing, we implement a reduced representation model and parallel-friendly algorithms within our frame budget. While the per-Gaussian-based discretization as done in PhysGaussian (Xie et al., 2023b) offers detailed dynamics, its time cost is impractical for our system's needs. Instead, we utilize a tetrahedral mesh embedding reconstructed from Gaussian kernels to drive GS motion. We adopt PBD with finite-element constraints to achieve real-time simulations.

Unified Framework VR-GS is a framework that integrates rendering and simulation within a unified pipeline. Our design principles adhere to the goal of "what you see is what you simulate" (Müller et al., 2016). Unlike Xie et al. (2023b), our system employs a hybrid approach, combining cage mesh and Gaussian kernels, to achieve real-time performance. The deformation gradient field of Gaussian kernels is embedded in the cage mesh through piecewise constant interpolation. With each Gaussian kernel resembling a rotated ellipsoid, this simple embedding strategy leads to spiky artifacts during large deformation, a challenge highlighted in (Xie et al., 2023b). To mitigate this, we employ a two-level interpolation scheme: initially embedding each Gaussian within an individual bounding tetrahedron, followed by embedding these tetrahedra within the simulated cage mesh to elevate the smoothness of the deformation field.

Additional Components Our framework is designed to construct VR-compatible interactive settings derived from real-world captures. Consequently, it introduces complexities

associated with facilitating intuitive user interactions and delivering real-time feedback that meets user expectations and common sense perceptions (Stanney et al., 2003; Hale and Stanney, 2014; Gabbard et al., 1999). In particular, when users engage with objects within a scene, they anticipate natural movements and realistic appearances from both the selected objects and the surrounding environment. Moreover, it is necessary to address voids when an object is removed from its supporting base. To manage these challenges, our system additionally integrates functionalities for object segmentation and scene inpainting.

6.3.4 Method

6.3.4.1 Gaussian Splatting

Gaussian splatting, proposed by Kerbl et al. (2023), is an explicit 3D representation to encapsulate 3D scene information using a set of 3D anisotropic Gaussian kernels, each with learnable mean μ, opacity σ, covariance matrix Σ, and spherical harmonic coefficients $\mathcal{C}$. Spherical harmonics (SH) are expansions of the view-dependent color function defined on the unit sphere. To render a view, the 3D splats are projected to 2D screen space ordered by their z-depth. The color C of a pixel is computed by α-blending of these 2D Gaussians from near to far:

$$C = \sum_{i \in N} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \tag{6.20}$$

where c_i is the evaluated color by SHs viewed from the camera to the kernel's mean. α_i is the product of the kernel's opacity and 2D Gaussian weight evaluated at the pixel coordinate. Leveraging a differentiable implementation, the rendering loss towards the ground truth image can be backpropagated to Gaussians' parameters for optimization.

In contrast to traditional NeRFs based on implicit scene representations, GS provides an explicit representation that can be seamlessly integrated with post-processing manipulations, such as animating and editing. Moreover, the efficient rasterization and superior rendering quality of 3D Gaussians facilitate their integration with VR.

6.3.4.2 VR-GS Assets Preparation

Each 3D asset in our VR system consists of a high-fidelity 3D GS reconstruction and an envoloping simulatable tetrahedral mesh in a moderate resolution to enable real-time physics-aware dynamics. These preparations are conducted offline before immersive and interactive editings within our VR environment, which includes:

- *Segmented GS Generation*: we support interactions with individual objects in a large scene, achieved by segmented GS reconstructions.
- *Inpainting*: The occluded parts between the objects and their supporting planes usually have no texture. We inpaint 3D GS representations leveraging a 2D inpainting technique.
- *Mesh Generation*: A simulation-ready tetrahedral mesh is generated for each object.

Segmentation The segmented GS is constructed during GS training. We first generate 2D masks on the multi-view RGB images by utilizing a 2D segmentation model (Cheng et al., 2024). Each segmented part is assigned a different color that is consistent across different views. Subsequently, we enhance the scene representation by integrating three additional learnable RGB attributes into the 3D Gaussian kernels. During the reconstruction process, each 3D Gaussian kernel will automatically learn what object it belongs to utilizing a segmentation loss function L_{seg}:

$$L_{\text{seg}} = L_1(M_{2d}, I), \tag{6.21}$$

where M_{2d} represents colored 2D segmentation results and I denotes the rendering of 3D Gaussian kernels with colors replaced by the extended RGB attributes instead of evaluated from SHs. Thus, the total loss becomes

$$L_{\text{total}} = (1 - \lambda)L_1 + \lambda L_{\text{SSIM}} + \lambda_{\text{seg}} L_{\text{seg}}, \tag{6.22}$$

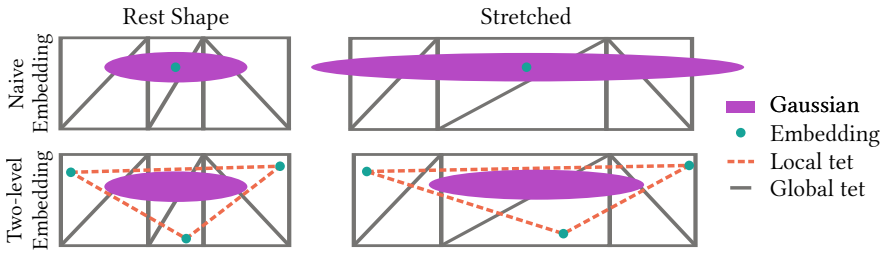


Figure 6.23: Our two-level embedding effectively resolves spiky artifacts. Each Gaussian kernel is embedded into a local tetrahedron. The vertices of the local tetrahedron are independently embedded into the global mesh.

where L_1 and L_{SSIM} are computed between the normally rendered images and the multi-view ground truths. We use $\lambda = 0.2$ and $\lambda_{\text{seg}} = 0.1$ in all our experiments.

Inpainting After 3D GS segmentation, we extract all objects separately from the scene. This process of object removal, however, results in the emergence of holes within the regions that are previously occluded. To alleviate the issue, we utilize a 2D inpainting tool LaMa (Suvorov et al., 2022) to guide the 3D inpainting of Gaussian kernels. We freeze the Gaussian kernels located outside of the holes and then use an inpainting loss $L_{\text{inpaint}} = L_1(I_{\text{inpainted}}, I)$ to optimize a Gaussian kernel patch under the guidance of the 2D inpainted images $I_{\text{inpainted}}$ for the current 3D GS rendering I. The result of our inpainting is validated in Section 6.3.5.

Mesh Generation Due to our design choice to use mesh-based simulation, we generate a tetrahedral mesh for each group of the segmented 3D GS kernels. These meshes will not be rendered during the interaction but only serve as the media of dynamics. Hence we can use a moderate mesh resolution which does not hinder performance. To construct a simulation mesh, we first use internal filling proposed by (Xie et al., 2023b) to fill particles into the void internal region that is not reconstructed by GS. Then we treat centers of Gaussians as a point cloud and convert it to a voxelized VDB representation (Museth, 2013). A water-tight surface mesh is then extracted using marching cubes (Lorensen and Cline, 1987) and tetrahedralized into a finite element mesh using TetGen (Si, 2015).

6.3.4.3 Unified Framework for Simulation and Rendering

As shown in Xie et al. (2023b), Gaussian kernels can be deformed by the simulated deformation field. We follow the Gaussian kinematics of this work but replace the simulator with XPBD (Macklin et al., 2016) to achieve real-time interactions. We employ the strain energy constraint as the elastic model and adopt the velocity-based damping model in the XPBD framework.

Physical Parameters The physical parameters, such as densities and material stiffness, are tunable hyperparameters. Following PhysGaussian (Xie et al., 2023b), we manually set physical parameters for all interactable objects, including Young's modules (E), Poisson ratio (ν), and density (ρ). We start from an initial configuration $E = 1000$ Pa, $\nu = 0.3, \rho = 1000$ kg for each object, then tune them to produce visually plausible dynamics. The Young's modulus governs the overall stiffness of an object and is manually adjusted by analyzing the severity of deformation of its free fall collision with the ground. Additionally, we adjust the relative density between two objects based on their deformations under collision. In practice we usually arrive at a finalized parameter setting that gives visually plausible results within 10 iterations of this manual process.

Embedding In our mesh-based simulation, the deformation map is piece-wise linear, with the resulting deformation gradient piece-wise constant within each tetrahedron. Given a tetrahedron with the rest-shape configuration $\{\boldsymbol{x}_0^0, \boldsymbol{x}_1^0, \boldsymbol{x}_2^0, \boldsymbol{x}_3^0\}$ and the current configuration $\{\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3\}$, the deformation gradient is defined as

$$\boldsymbol{F} = \left[\boldsymbol{x}_1 - \boldsymbol{x}_0, \boldsymbol{x}_2 - \boldsymbol{x}_0, \boldsymbol{x}_3 - \boldsymbol{x}_0\right] \left[\boldsymbol{x}_1^0 - \boldsymbol{x}_0^0, \boldsymbol{x}_2^0 - \boldsymbol{x}_0^0, \boldsymbol{x}_3^0 - \boldsymbol{x}_0^0\right]^{-1}, \tag{6.23}$$

where the inverse of the rest-shape basis can be computed pre-simulation. The mean and the covariance matrix of the deformed Gaussian kernel inside this tetrahedron is given by

$$\boldsymbol{\mu} = \sum_i w_i \boldsymbol{x}_i, \quad \boldsymbol{\Sigma} = \boldsymbol{F}\boldsymbol{\Sigma}_0\boldsymbol{F}^T, \tag{6.24}$$

where w_i is the barycentric coordinates of the initial center $\boldsymbol{\mu}_0$ in the rest-shape configuration, and $\boldsymbol{\Sigma}_0$ is the initial covariance matrix (Xie et al., 2023b). The deformed Gaussian kernels can be directly rendered by the point splatting procedure. However, the direct embedding of Gaussian centers cannot guarantee that every ellipse shape is completely enveloped by some tetrahedron inside the simulation mesh. As shown in Figure 6.23, this can lead to spiky artifacts. Observe that kernels that are completely inside the tetrahedron will always be inside. This motivates us to propose a two-level embedding procedure:

1. Local embedding: we independently envelope every Gaussian kernel by an as-tight-as-possible tetrahedron. There is no connectivity between these local tetrahedra.

2. Global embedding: we embed the vertices of local tetrahedra into the global simulation mesh.

As the global mesh is deformed by the simulation, the vertices of local tetrahedra are kept inside the boundary, driving the kinetic evolution of Gaussian kernels. A local tetrahedron could overlap with multiple global tetrahedra. The deformation map on it can be understood as the average of the surrounding global tetrahedra, hence eliminating sharp, spiky artifacts, as validated in Section 6.3.5.

Shadow Map While the original global illumination of the scene can be accurately learned and baked by the spherical harmonics on each Gaussian, the shadow will no longer be aligned with the object when it is moving or deforming. Bringing shadow map (Wanger et al., 1992) into the GS framework can enhance the immersive experience in VR. More importantly, it can guide human perception of the spatial relationships between objects: during manipulation, users will rely on the shadow to determine the distance between objects. The shadow map is a fast real-time algorithm and is well-aligned with the GS rasterization pipeline. We follow Equation (6.20) to estimate the depth map from the light source and test the visibility for each Gaussian using this depth map. The influence of shadow maps is studied in Section 6.3.5.

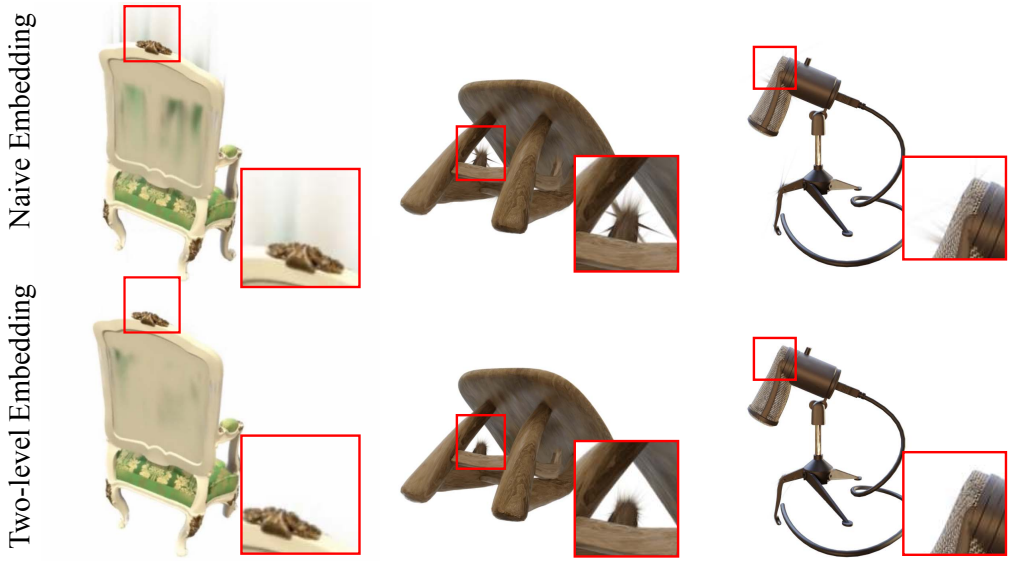


Figure 6.24: **Two-level Embedding Evaluation.** Our two-level embedding alleviates the commonly seen spiky artifacts for deformed GS kernels.

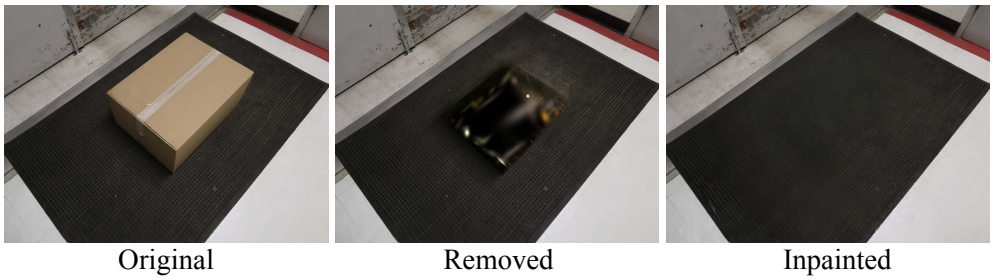


Figure 6.25: **Inpainting Evaluation.** GS struggles to reconstruct occluded surfaces. By leveraging LAMA (Suvorov et al., 2022), we produce 2D inpainted results that guide the 3D scene inpainting, enhancing the realism of the 3D representation.

6.3.5 Evaluation

In this section, we assess the essential components of our proposed system, including two-level GS embedding and shadow map.

Two-level Embedding The two-level embedding is a crucial component in our physical dynamics-aware system, integrating the tetrahedra cage mesh with the embedded Gaussian. In Figure 6.24, we conduct an ablation study to validate the effectiveness of our two-level embedding approach. Under conditions of extreme stretching or twisting, the naive embedding method, which simply embeds the Gaussian kernel within the closest tetrahedron, leads to

Figure 6.26: **Shadow Map Evaluation.** GS traditionally models shadows as static surface textures. Our approach, employing a shadow map, generates dynamic shadows for a more immersive experience.

severe spiky artifacts. Our two-level embedding strategy addresses this by initially embedding each Gaussian into a localized, independent tetrahedron, followed by embedding the vertices of this tetrahedron into the cage mesh. The deformation of each Gaussian kernel is determined by averaging the deformations at these vertices, resulting in a smoother deformation gradient field than the naive approach and significantly reducing spiky artifacts.

Inpainting With the guidance of 2D image segmentation results, our system achieves object-level 3D scene reconstruction, facilitating convenient post-physics-based manipulation. However, GS is limited to reconstructing surfaces visible in the provided multi-view training images. Consequently, some object and background areas, unseen in these images, remain unreconstructed by GS, leading to "black hole" like artifacts when foreground objects are moved. To address this, our system integrates the object segmentation mask with LAMA (Suvorov et al., 2022), a 2D inpainting model, to generate inpainted multi-view images. These images then guide the fine-tuning and inpainting of our 3D GS scene, yielding a more complete and realistic user experience, as evidenced in Figure 6.25.

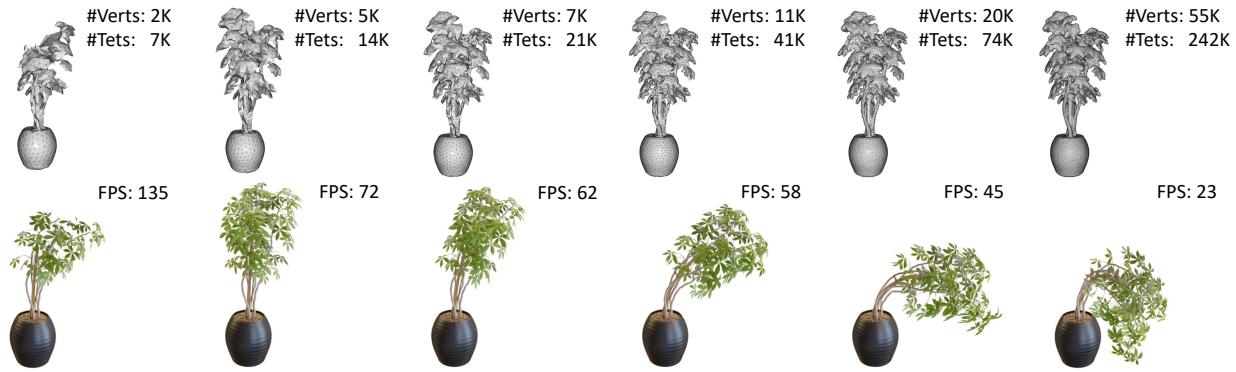


Figure 6.27: **Trade-offs between Quality and Performance.** The top row displays cage meshes at varying resolutions, and the bottom row illustrates the corresponding simulation dynamics. Low-resolution meshes fail to capture fine dynamic details, whereas high-resolution meshes compromise real-time performance and can result in overly soft artifacts in the simulated object due to non-converging simulations. We employ mid-resolution meshes in practice to achieve an optimal balance between high frame rates and realistic physical dynamics.

Shadow Map Furthermore, we take advantage of the shadow map to add extra time-dependent shadows into the GS scene, as shown in Figure 6.26. Original GS represents shadows as textures and thus fails to provide dynamic shadows when objects move. VR-GS allows users to choose the parameters, e.g. position and direction, of the light source to reproduce the lighting setting of the original scene, leading to a more realistic VR environment.

6.3.6 Experiment

In this section, we benchmark our VR-GS system's simulation performance against other methods in GS and NeRF manipulation. Additionally, we showcase interactive demonstrations on VR devices. Our prototype, developed in Unity with a CUDA-implemented plugin, is tested using a Quest Pro Head-Mounted Display (HMD) and corresponding controllers, on an Intel Core i9-14900KF CPU with 32GB memory and an NVIDIA GeForce RTX 4090 GPU.

6.3.6.1 Performance

VR-GS is designed for real-time, physics-aware interactions by integrating Gaussian kernels within a cage mesh for simulation. This cage mesh is derived from the VDB representation of

Figure 6.28: **Visual Quality Comparison.** Our method synthesizes competitive visual results compared to PhysGuassian (Xie et al., 2023b) and significantly outperforms PAC-NeRF (Li et al., 2023e).

Table 6.5: Quantitative comparisons.

Example	PAC-NeRF	Phys-Gaussian	VR-GS (**Ours**)
Stool	0.750	0.112	0.017
Chair	0.813	0.219	0.022
Materials	0.625	0.39	0.021

Gaussians, with adjustable mesh resolution to influence quality. We first conduct experiments to explore the trade-offs between mesh quality and system performance. As shown in Figure 6.27, using a coarse cage mesh facilitates higher frame rates but may lead to the loss of fine details. Conversely, a higher-resolution mesh inevitably increases the computational cost. Moreover, PBD also requires much more iterations to achieve convergence for finer meshes. Without sufficient iterations, the simulated object may appear to be overly soft, yielding unrealistic dynamics. In practice, we constrain the number of mesh vertices to between 10K and 30K to maintain a balance between high frame rates and accurate physical dynamics.

We then compare VR-GS against two state-of-the-art NeRF/GS physics-based manipulation methods: PAC-NeRF (Li et al., 2023e) and PhysGaussian (Xie et al., 2023b). PAC-NeRF primarily concentrates on estimating material parameters from multi-view videos in recon-

Table 6.6: Parameters and Timings of Demos in Section 6.3.6.2.

Example	#Gaussians	#Verts.	#Iter*	CD*	FPS
(Figure 6.30) Fox	304,875	28,205	1	/	161.2
(Figure 6.30) Bear	2,525,891	19,472	5	/	76.9
(Figure 6.30) Horse	1,295,223	10,611	5	/	115.4
(Figure 6.31) Ring Toss	1,456,209	9,002	10	10	37.7
(Figure 6.31) Table Brick Game	1,866,835	33,279	10	10	41.4
(Figure 6.32) Toy Collection	1,665,128	46,428	20	20	24.3
(Figure 6.25) Box Moving	1,769,412	1,434	10	/	73.8
(Figure 6.21) Animal Crossing	1,312,670	36,386	10	5	34.7
(Figure 6.33) Just Dance	1,059,054	13,936	50	/	33.9

* #Iter: XPBD iterations per substep; CD: collision detections per step.

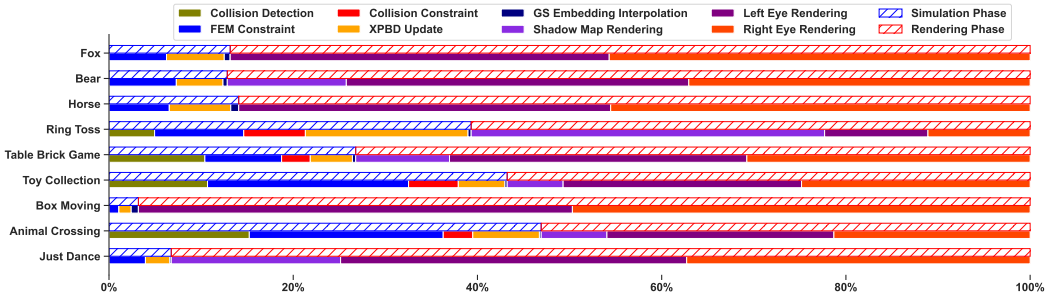


Figure 6.29: **Timing Breakdown of Demos in Section 6.3.6.2.**

structed NeRF scenes. Although it offers novel dynamic generation capabilities, the resulting visuals often fall short in rendering quality. In contrast, PhysGaussian, leveraging GS, produces superior photorealistic dynamics. However, the Material Point Method (MPM) used by both systems can limit real-time performance in complex scenes. For a fair comparison, we standardize the frame time ($\Delta t_{\text{frame}} = 1/25$ sec) and the simulation step time ($\Delta t_{\text{step}} = 0.0001$ sec) across all methods. For VR-GS, we set the XPBD iteration per substep to 1, as the small Δt_{step} sufficiently resolves the dynamics. As depicted in Figure 6.28 and Table 6.5, VR-GS not only matches the visual quality of PhysGaussian but also outperforms PAC-NeRF in clarity and realism. Crucially, our PBD-based simulation framework allows for significantly faster simulations, making VR-GS ideal for real-time physics-aware interactions.

6.3.6.2 VR Demos

We then showcase VR-GS's capability to replicate real-world scenarios through several representative demos in VR. Detailed simulation setups and a timing breakdown for these

Figure 6.30: **Fox, Bear, and Horse**. VR-GS allows users to manipulate 3D GS at a real-time rate with physically plausible responses.

demos are provided in Table 6.6 and Figure 6.29.

Fox, Bear, and Horse Manipulation VR-GS empowers users to edit 3D Gaussian kernels intuitively and efficiently, utilizing our XPBD-based physics engine for dynamic manipulation. We demonstrate this through three examples as depicted in Figure 6.30: a fox, a bear, and a horse, reconstructed from the Instant-NGP dataset (Müller et al., 2022), the Instruct-NeRF2NeRF (Haque et al., 2023), and the Tanks and Temples dataset (Knapitsch et al., 2017), respectively.

Ring Toss and Table Brick Game In this example, we demonstrate the system's ability to seamlessly integrate new *virtual objects* into existing scenes. We use a room scene reconstructed from real-world footage and virtual objects (rings and bricks modeled in Blender) to create ring tossing and table brick game (Figure 6.31). The dining table and objects on it serve as collision boundaries for the elastic ring and rigid bricks (Deul et al.,

Figure 6.31: **Ring Toss and Table Brick Game.** Participants can engage in interactive ring toss and table brick breakout games simulated within authentic environments.

Figure 6.32: **Toy Collection.** We place all the toys on the ground first, and then move them into the basket.

2016).

Toy Collection and Animal Crossing We reconstructed a living room scene and a set of plush toys (Figure 6.32) for users to interact with. The yellow spherical toy is generated using a text-to-3D generator LucidDreamer (Liang et al., 2023), while the others are reconstructed. Additionally, four animal plush toys were placed on chairs, allowing users to manipulate and deform them freely (Figure 6.21).

Just Dance VR-GS also supports animation of reconstructed GS humans (Figure 6.33). Utilizing a reconstructed human character, we leverage Mixamo[1]'s auto-rig to generate motion sequences of its surface mesh. These sequences then serve as the boundary condition for the simulation, finally producing the GS human animation that can be viewed in our immersive system.

[1]https://www.mixamo.com/

Figure 6.33: **Just Dance.** VR-GS generates high-fidelity dance given a reconstructed human body.

6.3.6.3 Modeling Details

We employ the *Toy Collection* (Figure 6.32) as a case study to demonstrate the time and labor required to create a fully interactive VR environment from scratch. The setup process is as follows:

1. The initial stage involves setting up the real-world scene. For the *Toy Collection*, the indoor scene was captured directly as one scene, with all toys separately hung with strings as another scene. We completed it within 20 minutes.

2. Next, we capture multi-view images for Gaussian Splatting by recording videos of the two scenes with a consumer-level camera and converting them into image sequences. Camera intrinsic and extrinsic parameters are collected using COLMAP, with the entire process taking around 60 minutes, of which COLMAP occupies approximately 40 minutes.

3. We then proceed to image processing for segmentation and inpainting. Object classes are manually designated in the first video frame. Subsequent frames are automatically segmented using the method from Cheng et al. (2024), taking about 2 minutes. The inpainting of what's below the basket is fully automated (Suvorov et al., 2022) and requires 5 minutes.

4. GS reconstruction is performed on the processed images, undergoing 30,000 training steps as specified by Kerbl et al. (2023) without additional treatment for the number of Gaussians, which takes about 30 minutes on a single RTX 4090.

5. After generating 3DGS, we execute internal filling, VDB reconstruction, and TetGen,

all within 10 minutes.

6. Lastly, object world positions and physical parameters are specified and estimated in Unity, as detailed in Section 6.3.4.3, taking 20 minutes.

In summary, the total preparation time for the *Toy Collection* is approximately 2 hours and 30 minutes. Manual tasks include scene staging, video recording, object class specification, and physical parameter selection. All other processes are automated. Once modeled, the scene is ready for a physics-aware interactive VR experience.

6.3.7 User Study

We conducted a user study including two tasks with 10 participants: 2 professionals (P1–P2) with 5 and 7 years of experience in 3D VFX software (including Houdini and Blender), and 8 novices (P3–P10), with P2 also having 2 years of VR experience. Our study used the hardware specified in the experimental section and included the Fox, Bear, House, Ring Toss, and Toy Collection demos. Participants received a 10-minute video tutorial on our system before completing two tasks. After task 2, they answered a questionnaire covering usability (System Usability Scale (Bangor et al., 2009)) and subjective feedback on individual and overall system features, as shown in Section 6.3.7.2. Additionally, we conducted a 20-minute semi-structured interview for more in-depth feedback.

6.3.7.1 Tasks

Task 1: Goal-directed (30 minutes). The participants were required to play two VR games developed by our system: ring toss and toy collection (Figure 6.32) in two different settings (S1: physics-based and S2: transform-based interaction). Each game had a 5-minute duration under each configuration. A 5-minute break was allowed between sessions. The arrangement of sessions and tasks used a Latin square design to counter potential learning effects. We requested users to rate the immersive and realistic experience in VR using a 7-point Likert scale.

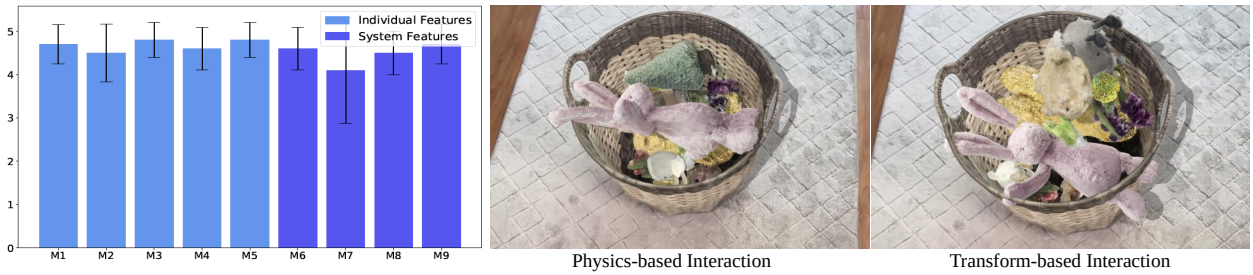


Figure 6.34: **User Study Results.** The left chart summarizes individual (M1-M5: object manipulation, scene inpainting, illumination, dynamics, physics placement) and system feedback (M6-M9: ease of use, latency, functionality, satisfaction). The right two figures show that physics-based interaction enhances immersion and realism in editing, whereas transform-based interaction yields less authentic outcomes, e.g. an undeformed and penetrated toys.

Task 2: Open-ended (30 minutes). The participants are asked to edit the scene and generate dynamics freely in aforementioned scenes (shown in Figure 6.30), Figure 6.31 and Figure 6.32) with our prototype system, which includes geometry-based editing, physics-based editing, transform, rotation, duplicating, undoing, rescaling, etc. Task 2 is designed to evaluate the detailed potential factors impacting VR immersive experience.

6.3.7.2 Results and Discussion

According to a paired t-test conducted on the score collected from task 1, it could be noticed that physics-based interaction significantly enhanced user immersion and realism compared to transform-based interaction (physics-based: 6.1 vs transform-based: 4.8 on average, $p = .0227$). As shown in Figure 6.34, our system has received overall positive opinions. Users spoke highly of the placement of virtual content. Transforming objects without physics rules resulted in floating objects, while in our system, all virtual content placements, such as putting toys onto a sofa, are followed by physics rules and are consistent with the user's understanding. P7 commented, *"I love this a lot. I have a dog. Petting the fox is really like what I did to my dog at home. I felt so real. Moreover, the placement of the toy is awesome. After they all fell to the basket, the basket even shook for a while."* The VR-GS system's realistic lighting and physics-based dynamics create an immersive experience. As a professional user, P2 spoke highly of high-fidelity generative dynamics and illumination. *"Those motions of virtual content are just like what I see in the physical world. I can't wait*

to take photos of my own house and then put them in VR with my video game character." Users rated the system highly on ease of use (4.6/5) and overall satisfaction (4.8/5). With a System Usability Scale (SUS) score of 83.5, VR-GS is classified as "excellent" according to (Bangor et al., 2009).

6.3.8 Conclusion and Future Work

We presented a physical dynamics-aware interactive Gaussian Splatting system for addressing challenges in editing real-time high-fidelity virtual content. By leveraging the advancements in Gaussian Splatting, VR-GS bridges the quality gap traditionally observed between machine-generated and manually created 3D content. Our system not only enhances the realism and immersion via physically-based dynamics, but also provides fine-grained interaction and manipulation controllability.

Although all the study participants appreciated the efficiency and effectiveness of our system, it still remains to be improved. Firstly, rendering high-fidelity Gaussian kernels in VR is computationally demanding. As a result, rendering generative dynamics in a large scene with 2K resolution might lead to potential latency issues in our system. Secondly, the physical parameters in our system are manually defined. Estimating parameters from videos like PAC-NeRF (Li et al., 2023e), or leveraging large-vision models, would be interesting directions to automate this process. In future work, we aim to incorporate a broader range of materials, such as fluid (Feng et al., 2024) and cloth, to enhance the system's capabilities. Furthermore, it is also interesting to explore how to utilize large multimodal models to assess the fidelity of the generated dynamics.

6.4 Dress-1-to-3: Single Image to Simulation-Ready 3D Outfit with Diffusion Prior and Differentiable Physics

6.4.1 Introduction

Creating digital assets of clothed humans is crucial for a wide range of applications, including virtual reality (VR), the film industry, fashion design, and gaming. However, the traditional pipeline for digital human and garment creation involves multiple intricate steps, such as concept design, material selection, garment modeling, human pose generation, garment fitting, and animation. These processes are often labor-intensive and time-consuming.

In recent years, significant advancements in image-to-3D asset reconstruction have been driven by the development of powerful image and video generation models. Among these, multiview diffusion models (Chen et al., 2024d; Liu et al., 2023b; Gao et al., 2024) have emerged as a promising approach, effectively leveraging multiview images as intermediate representations to capture 3D information. When fine-tuned on human datasets, these models generalize well to avatar reconstructions from in-the-wild images (Li et al., 2024b; He et al., 2024b). However, the generated results are often fused into a single piece, making them unsuitable for downstream tasks such as garment animation and interaction.

In the meantime, sewing patterns, a foundational representation in the garment design industry, have been adopted as intermediate reconstruction outputs to recover garment geometries (Liu et al., 2023a; Li et al., 2024c). This representation is particularly advantageous due to its seamless integration with downstream applications such as physics simulation and garment editing. Despite their promise, these feed-forward approaches face significant limitations stemming from the scarcity of high-quality 3D data. As a result, the reconstructed garments are often constrained by the distribution of the training dataset, leading to inaccuracies in aligning with input images. This limitation hinders their ability to produce detailed and diverse reconstructions reflective of real-world garment variations. The question then arises: can we keep the advantages of the simulation-ready representation of sewing patterns while leveraging the powerful priors in large multi-view diffusion models to reconstruct garments

Figure 6.35: Dress-1-to-3 can reconstruct simulation-ready textured clothed humans from casually posed single view images.

from solely an in-the-wild image?

To address this problem, we introduce **Dress-1-to-3**, a novel garment reconstruction pipeline that accurately transforms an in-the-wild image into a simulation-ready representation of separated human and garment by leveraging the strengths of both 2D multi-view diffusion and 3D sewing pattern reconstruction. To bridge those two parts, we propose a generalized and unified IPC differentiable framework for garment optimization, which enables the optimization of 3D sewing patterns using 2D generative multi-view RGB images and normal maps as guidance. By refining imperfect generative outputs to align with the geometry encoded in multiview images, our approach allows the reconstruction of out-of-distribution garment shapes with high fidelity. Our contributions include:

- We propose a holistic garment reconstruction pipeline that takes a single image as input and generates garments fitted onto a posed human, ensuring both the human pose and garments align with the input image.
- We derive a generalized and unified IPC differentiable framework that is agonistic to constitutive models. We apply this framework for co-dimensional garment optimization.

- We conduct extensive experiments to demonstrate the effectiveness and versatility of our garment optimization framework, successfully reconstructing garments across diverse categories, including those not present in the training dataset.

6.4.2 Related Work

Multi-view Diffusion Owing to their powerful predictive ability, Diffusion Probabilistic Models (Ho et al., 2020) have been applied to image (Nichol et al., 2021; Zhang et al., 2023; Dhariwal and Nichol, 2021; Ruiz et al., 2023; Saharia et al., 2022), video (Chen et al., 2024c; Ho et al., 2022), and 3D shape synthesis tasks (Long et al., 2024; Yu et al., 2023; Tang et al., 2023), etc. However, applying image diffusion models to generate multi-view images separately poses significant challenges in maintaining consistency across different views. To address multi-view inconsistency, multi-view attentions and camera pose controls are adopted to fine-tune pre-trained image diffusion models, enabling the simultaneous synthesis of multi-view images (Shi et al.; Wang and Shi, 2023; Xu et al., 2023d; Yang et al., 2024; Shi et al., 2023; Long et al., 2024), though these methods might result in compromised geometric consistency due to the lack of inherent 3D biases. To ensure both global semantic consistency and detailed local alignment in multi-view diffusion models, 3D-adapters (Chen et al., 2024b) propose a plug-in module designed to infuse 3D geometry awareness. Nevertheless, the generated images by these models are sparse views. To address this issue, CAT3D (Gao et al., 2024) introduces an efficient parallel sampling strategy to generate a large set of camera poses, and MVDiffusion++ (Tang et al., 2025) adopts a pose-free architecture and a view dropout strategy to reduce computational costs, generating dense, high-resolution images.

Generating consistent images from multi-view diffusions offers guidance for further 3D shape reconstruction (Gao et al., 2024). PSHuman (Li et al., 2024b) integrates a body-face cross-scale diffusion with an SMPL-X conditioned multi-view diffusion for clothed human reconstruction with high-quality face details. Recent work, MagicMan (He et al., 2024b), utilizes a hybrid human-specific multi-view diffusion model with 3D SMPL-X-based body priors and 2D diffusion priors to consistently generate dense multi-view RGB images and

normal maps, supporting high-quality human mesh reconstruction. Different from these works, we exploited multi-view diffusions to generate multi-view normals and RGB images as guidance to optimize sewing patterns and stitches instead of human meshes.

Garment Reconstruction Previous work focusing on clothed human reconstruction (Xiu et al., 2022, 2023) typically generates garments fused with digital human models, limiting them to basic skinning-based animations and requires extra segmentation and editing to separate the garments from the human body. In contrast, our approach focuses on reconstructing separately wearable, simulation-ready garments and human models. Other closely related works include Li et al. (2024d), which also generates simulation-ready clothes, but at the cost of creating clothing templates by artists and precise point clouds by scanners. NeuralTailor (Korosteleva and Lee, 2022) exploits point-level attention for pattern shape and stitching information regression, enabling the reconstruction of garment meshes from point clouds. In contrast, our paper focuses on reconstructing non-watertight garments and humans separately from a single image without additional inputs.

To reconstruct separated non-watertight garments from a single image, GarVerseLOD (Luo et al., 2024) recovers garment details hierarchically in a coarse-to-fine framework. However, it fails to reconstruct complex skirts or dresses with slits or with complex human poses due to the limited representation of such features in the training data. ClothWild (Moon et al., 2022) exploits a weakly supervised pipeline with DensePose-based loss to further increase robustness on in-the-wild images. BCNet (Jiang et al., 2020a) introduces a layered garment representation and a generic skinning weight generation network to model garments with different topologies. Deep Fashion3D (Zhu et al., 2020) refines adaptable templates with rich annotations to fit garment shapes. While they are limited to garment categories in their training datasets, these works fail to reconstruct complex categories such as jumpsuits. Additionally, they require nearly frontal images as input, limiting reconstruction from different views. AnchorUDF (Zhao et al., 2021) explores a learnable unsigned distance function to query both 3D position features and pixel-aligned image features via anchor points, which reconstructs the coarse garment shape but lacks the generation of high-quality geometric

details.

Instead of directly reconstructing garment meshes, some works (Liu et al., 2023a; He et al., 2024a; Korosteleva and Sorkine-Hornung, 2023) treat sewing patterns as intermediate representations to generate garments by stitching them together. Recent work, GarmentRecovery (Li et al., 2024c), introduces implicit sewing patterns (ISP) to provide shape priors integrated with deformation priors for further garment recovery, though it builds specialized models for each individual garment or garment type. Both SewFormer (Liu et al., 2023a) and PanelFormer (Chen et al., 2024a) utilize Transformers to predict sewing patterns and stitches. However, their garment results lack physical material parameters. Therefore, they fail to reconstruct diverse shapes for garments with different physical materials. In addition, these feed-forward methods require large amounts of garment data for training, failing to synthesize garments that fall outside the distribution of the training data. Our work aims to generate diverse, image-aligned, simulation-ready garments with high-quality details from in-the-wild images by optimizing sewing patterns and stitches with physical parameters via differentiable simulations.

Differentiable Simulation Differentiable simulation has seen widespread application in recent research, particularly for system identification and the inference of material parameters from both synthetic (Li et al., 2023e, 2024e) and real-world (Huang et al., 2024; Si et al., 2024) observations. The scope of exploration spans various domains, including fluid dynamics and control (McNamara et al., 2004; Schenck and Fox, 2018; Li et al., 2023g, 2024e), rigid-body dynamics (Freeman et al., 2021; Strecke and Stueckler, 2021; Xu et al., 2023a), articulated systems (Geilinger et al., 2020; Qiao et al., 2021b; Xu et al., 2021), soft-body dynamics (Hahn et al., 2019; Hu et al., 2019b; Du et al., 2021; Jatavallabhula et al., 2021; Huang et al., 2024), cloth (Li et al., 2022h; Stuyck and Chen, 2023; Li et al., 2024d), inelasticity (Huang et al., 2021; Li et al., 2023e), inflatable structures (Panetta et al., 2021), and Voronoi diagrams (Numerow et al., 2024).

Cloth-based applications, whether for static optimization or dynamic simulation, frequently involve extensive frictional contact. Consequently, many works focus on robust methods for

handling dry frictional contact in differentiable simulations. Bartle et al. (2016) proposes a physics-driven pattern adjustment for garment editing using fixed-point optimization, which does not account for gradients. Liang et al. (2019) is the first to introduce a fully functional differentiable cloth simulator with frictional contact and self-collision, formulating a quadratic programming problem. Jatavallabhula et al. (2021) employs a penalty-based frictional contact model, while Du et al. (2021) and Li et al. (2022h) leverage the adjoint method for Projective Dynamics (Bouaziz et al., 2014) with friction. Building on Position-Based Dynamics (Müller et al., 2007; Macklin et al., 2016), Stuyck and Chen (2023) and Li et al. (2024d) introduce differentiable formulations for compliant constraint dynamics, and Huang et al. (2024) presents an adjoint-based framework for differentiable Incremental Potential Contact (IPC) (Li et al., 2020, 2021a).

The finite difference (FD) method (Renardy and Rogers, 2006) is a standard approach to numerical differentiation. The complex-step finite difference technique (Luo et al., 2019; Shen et al., 2021) offers an alternative that mitigates issues such as subtractive cancellation and accumulated numerical errors by leveraging complex Taylor expansions (Brezillon et al., 1981). They can be used to optimize low-DoF system (Zheng et al., 2025). Automatic differentiation (AD) (Naumann, 2011; Margossian, 2019) and code transformation libraries like NVIDIA Warp (Macklin, 2022), DiffTaichi (Hu et al., 2019b, 2020), and others (Herholz et al., 2024) automatically compute gradients based on forward simulation, allowing for greater reuse of existing code. However, they can introduce code constraints, incur a high memory footprint, and may cause gradient explosion if applied naively. Our framework combines NVIDIA Warp's AD with an adjoint method to achieve both development efficiency and high performance.

6.4.3 Differentiable Garment Simulation

6.4.3.1 Forward Simulation

We use Codimensional Incremental Potential Contact (CIPC) (Li et al., 2021a) as our underlying garment simulation method, which is the state-of-the-art in cloth simulation

regarding accuracy and robustness. It ensures non-penetration through distance-based log barrier energy and continuous collision detection (CCD). Below, we summarize the simulation pipeline, with further details available in Li et al. (2021a).

The simulated codimensional surface is discretized into triangles defined by vertices $\boldsymbol{V}$ and faces $\boldsymbol{F}$. Let $\boldsymbol{X}$ denote the vertex positions in the undeformed state, and let $\boldsymbol{x}^n$ and $\boldsymbol{v}^n$ represent the vertex positions and velocities, respectively, at time step t^n. CIPC employs an optimization-based time integrator to achieve the state transition from time step t^n to $t^{n+1} = t^n + h$, minimizing the following energy:

$$\boldsymbol{x}^{n+1} = \arg\min_{\boldsymbol{x}} E(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_{\boldsymbol{M}}^2 + \Psi(\boldsymbol{x}; \boldsymbol{X}) + B(\boldsymbol{x}). \tag{6.25}$$

Here, $\tilde{\boldsymbol{x}} = \boldsymbol{x}^n + \boldsymbol{v}^n h + \boldsymbol{g}h^2$ represents the predictive position under backward Euler integration. $\|\cdot\|_{\boldsymbol{M}}$ denotes the L^2-norm weighted by the vertex mass $\boldsymbol{M}_{ii}$. $\Psi(\boldsymbol{x}; \boldsymbol{X})$ is the elastic energy, encompassing both stretching and bending energies, depending on the user's choice. $B(\boldsymbol{x})$ is the log barrier energy introduced by IPC, defined over all contacting vertex-triangle and edge-edge pairs. The barrier energy for each pair of primitives increases from zero to infinity as the gap decreases from a threshold $\hat{d}$ to 0, providing sufficient repulsion to prevent penetrations.

Newton's method with line search is employed to solve the optimization problem, requiring the analytical computation of the gradient and Hessian matrix of the energy at each iteration. The step size upper bound in each line search is clamped by CCD to ensure that all intermediate states remain intersection-free, provided that $\boldsymbol{x}^n$ is initially intersection-free. Finally, the new velocity is updated as $\boldsymbol{v}^{n+1} = (\boldsymbol{x}^{n+1} - \boldsymbol{x}^n)/h$.

6.4.3.2 Differentiable CIPC

Huang et al. (2024) provided an analytical derivation of differentiable IPC using the adjoint method. However, their derivation is closely tied to specific choices of constitutive models. To extend their framework to support cloth simulation, tedious derivations of analytical derivatives are required. In this work, we present a simple and unified framework that leverages both automatic differentiation and the adjoint method.

The governing equation of CIPC simulation can be expressed as an implicit nonlinear system of equations derived from the first-order optimality condition of the minimizer for Equation (6.25):

$$\boldsymbol{G}(\boldsymbol{x}^*; \boldsymbol{x}^n, \boldsymbol{v}^n, \boldsymbol{\varsigma}^n) = \nabla E(\boldsymbol{x}^*; \boldsymbol{x}^n, \boldsymbol{v}^n, \boldsymbol{\varsigma}^n) = \boldsymbol{0}, \tag{6.26}$$

$$\boldsymbol{x}^{n+1} = \boldsymbol{x}^*, \quad \boldsymbol{v}^{n+1} = \frac{1}{h}(\boldsymbol{x}^* - \boldsymbol{x}^n), \tag{6.27}$$

Here, $\boldsymbol{x}^*$ is the minimizer of the system energy E, $\boldsymbol{x}^n, \boldsymbol{v}^n$ represents the last system state, and $\boldsymbol{\varsigma}^n$ denotes the set of all continuous parameters of the implicit equation, including shape parameters $\boldsymbol{X}$, mass matrix $\boldsymbol{M}$, elastic moduli, and others. We assume $\boldsymbol{\varsigma}^n$ are independent, although they may share the same values. This abstraction allows the simulator to function as a differentiable layer with $\boldsymbol{x}^n, \boldsymbol{v}^n, \boldsymbol{\varsigma}^n$ as input and $\boldsymbol{x}^{n+1}, \boldsymbol{v}^{n+1}$ as output. The computational graph can be handled by any auto-differentiable framework such as PyTorch. The backward operator computes $\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{x}^n}$, $\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{v}^n}$, and $\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\varsigma}^n}$ given $\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{x}^{n+1}}$ and $\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{v}^{n+1}}$ for a given training loss function $\mathcal{L}$.

Taking the full derivatives of Equation (6.26) with respect to $\boldsymbol{x}^n, \boldsymbol{v}^n, \boldsymbol{\varsigma}^n$ on both sides, we obtain:

$$\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{x}^*}\left[\frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{x}^n}, \frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{v}^n}, \frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{\varsigma}^n}\right] + \left[\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{x}^n}, \frac{\partial G}{\partial \boldsymbol{v}^n}, \frac{\partial G}{\partial \boldsymbol{\varsigma}^n}\right] = \boldsymbol{0}, \tag{6.28}$$

which leads to

$$\left[\frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{x}^n}, \frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{v}^n}, \frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{\varsigma}^n}\right] = -\left[\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{x}^*}\right]^{-1}\left[\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{x}^n}, \frac{\partial G}{\partial \boldsymbol{v}^n}, \frac{\partial G}{\partial \boldsymbol{\varsigma}^n}\right]. \tag{6.29}$$

By the chain rule, we have:

$$\begin{aligned}\left[\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{x}^n}, \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{v}^n}, \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\varsigma}^n}\right] = &\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{x}^{n+1}}\left[\frac{\mathrm{d}\boldsymbol{x}^{n+1}}{\mathrm{d}\boldsymbol{x}^n}, \frac{\mathrm{d}\boldsymbol{x}^{n+1}}{\mathrm{d}\boldsymbol{v}^n}, \frac{\mathrm{d}\boldsymbol{x}^{n+1}}{\mathrm{d}\boldsymbol{\varsigma}^n}\right] \\ &+\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{v}^{n+1}}\left[\frac{\mathrm{d}\boldsymbol{v}^{n+1}}{\mathrm{d}\boldsymbol{x}^n}, \frac{\mathrm{d}\boldsymbol{v}^{n+1}}{\mathrm{d}\boldsymbol{v}^n}, \frac{\mathrm{d}\boldsymbol{v}^{n+1}}{\mathrm{d}\boldsymbol{\varsigma}^n}\right].\end{aligned} \tag{6.30}$$

Here, we assume $\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{x}^n}$, $\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{v}^n}$, and $\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\varsigma}^n}$ are all row vectors to ensure dimension consistency. From

Equation (6.27), we have:

$$\mathrm{d}\boldsymbol{x}^{n+1} = \mathrm{d}\boldsymbol{x}^*, \quad \mathrm{d}\boldsymbol{v}^{n+1} = \frac{1}{h}(\mathrm{d}\boldsymbol{x}^* - \mathrm{d}\boldsymbol{x}^n). \tag{6.31}$$

Plugging Equation (6.31) into Equation (6.30), we obtain:

$$\begin{aligned}\left[\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{x}^n}, \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{v}^n}, \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\varsigma}^n}\right] = \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{x}^{n+1}} \left[\frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{x}^n}, \frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{v}^n}, \frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{\varsigma}^n}\right] \\ + \frac{1}{h}\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{v}^{n+1}} \left[\frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{x}^n} - \boldsymbol{I}, \frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{v}^n}, \frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{\varsigma}^n}\right].\end{aligned} \tag{6.32}$$

With some rearrangements, we arrive at:

$$\begin{aligned}\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{x}^n} &= \left[\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{x}^{n+1}} + \frac{1}{h}\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{v}^{n+1}}\right]\frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{x}^n} - \frac{1}{h}\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{v}^{n+1}} \\ \left[\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{v}^n}, \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\varsigma}^n}\right] &= \left[\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{x}^{n+1}} + \frac{1}{h}\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{v}^{n+1}}\right]\left[\frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{v}^n}, \frac{\mathrm{d}\boldsymbol{x}^*}{\mathrm{d}\boldsymbol{\varsigma}^n}\right].\end{aligned} \tag{6.33}$$

Denote $\mathcal{A} = \left[\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{x}^{n+1}} + \frac{1}{h}\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{v}^{n+1}}\right]\left[\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{x}^*}\right]^{-1}$. By Equation (6.29), we have:

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{x}^n} = -\mathcal{A}\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{x}^n} - \frac{1}{h}\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{v}^{n+1}}, \tag{6.34}$$

$$\left[\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{v}^n}, \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\varsigma}^n}\right] = -\mathcal{A}\left[\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{v}^n}, \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{\varsigma}^n}\right]. \tag{6.35}$$

Observe that $\mathcal{A}$ is obtained by solving a linear system, where the coefficient matrix $\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{x}^*}$ is the Hessian matrix of the system energy E. The term $\mathcal{A}\left[\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{x}^n}, \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{v}^n}, \frac{\partial \boldsymbol{G}}{\partial \boldsymbol{\varsigma}^n}\right]$ back-propagates the differentials in $\mathcal{A}$ to $\boldsymbol{x}^n$, $\boldsymbol{v}^n$, and $\boldsymbol{\varsigma}^n$ through $\boldsymbol{G}$, respectively. This process can be implemented by treating $\boldsymbol{G}$ as a differentiable layer that supports auto-differentiation. Using AutoDiff, we eliminate the need to manually derive the analytical expressions for $\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{v}^n}$ and $\frac{\partial \boldsymbol{G}}{\partial \boldsymbol{\varsigma}^n}$. All other components required for forward simulations have already been derived.

6.4.4 Method Overview

We start our pipeline by estimating an initial garment sewing pattern from a single-view image. Next, we generate consistent multi-view RGB images and their corresponding normal

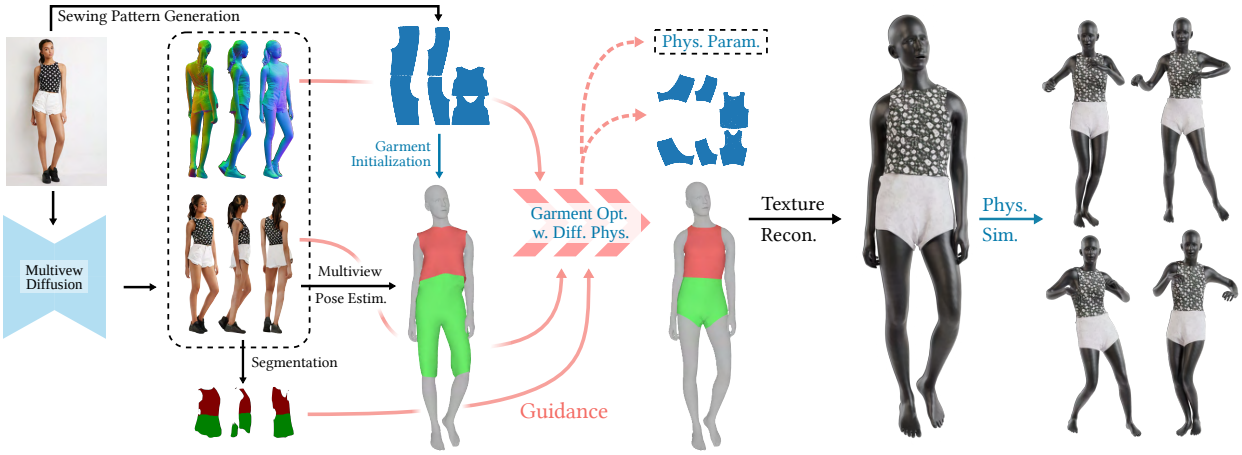


Figure 6.36: **Dress-1-to-3 Pipeline.** Starting with a single-view input image of a clothed human, we first derive an initial estimation of the sewing pattern. Additionally, we employ multi-view diffusion to generate orbital camera views, which serve as ground-truth 3D information for both human pose and garment shape. Next, we utilize differentiable simulation to sew and drape the pattern onto the posed human model, optimizing its shape and physical parameters in conjunction with geometric regularizers. Finally, the optimized garment shape provides a physically plausible rest shape in its static state and is readily animatable using a physical simulator.

maps, based on which we predict the human body pose. The 3D garment is initialized by stitching and draping the 2D patterns onto the predicted human model. The garment's interaction with the human body is simulated using a differentiable CIPC simulator, allowing us to optimize the physical parameters and the shapes of the sewing patterns guided by the previously generated multi-view RGB images, normal maps, and segmentation results. The optimized state produces a simulation-ready scene with a human model wearing well-fitted 3D outfits that align with the input. Garment textures are automatically generated using a visual-language model and image diffusion. Finally, by applying our CIPC simulator, we can simulate dynamic scenes where the predicted human body wears the optimized garments while performing various motion sequences. An illustration of the pipeline is shown in Figure 6.36. We elaborate on each component of the pipeline in the following sections.

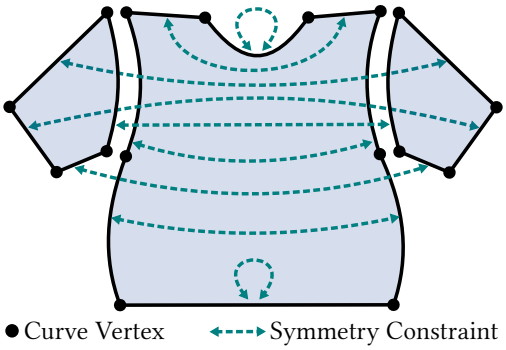


Figure 6.37: Illustration of symmetrization constraints.

6.4.5 Pre-Optimization Steps

6.4.5.1 Simulatable Sewing Pattern Generation

From a single-view image, our pipeline starts by generating an initial sewing pattern decomposition along with stitch information using SewFormer (Liu et al., 2023a). Following SewFormer's convention, the sewing pattern is represented as a set of quadratic Bézier curves on a 2D plane, forming a collection of disconnected patches. The curves for each patch are connected to form a loop. Let $\mathcal{E}$ denote the set of all curves, with its control parameters comprising the set of curve vertices $\mathcal{P} = \{\boldsymbol{P}_i\}$ and the set of control points $\mathcal{K} = \{\boldsymbol{K}^e\}$ for each edge curve $e \in \mathcal{E}$. To enable garment simulation, the patches are discretized into triangle meshes. First, we apply arc-length parameterization to achieve uniform sampling along the patch boundaries. For stitched patch edges, we ensure they share the same number of sampled points. This consistency allows us to apply vertex-to-vertex stitch constraints in garment simulations, simplifying the sewing process. Using the sampled boundary points, we then perform Delaunay triangulation (Shewchuk, 2008) independently for the interior of each patch.

Patch Symmetrization The sewing patterns generated by SewFormer often display certain symmetries, which we aim to preserve during garment optimization. SewFormer generates a fixed number of patches with a predefined order for patch names, though some patches may remain inactive. Symmetry information, including self-symmetry and inter-symmetry, is

embedded in these patch names. Symmetric edge pairs can be automatically identified by overlapping a patch with its mirrored symmetric counterpart or, in the case of self-symmetry, with the mirrored version of itself. Given the set of symmetric edge pairs $\mathcal{E}_S = \{(i,j) \sim (k,l)\}$, we define the *validated* curve vertices $\{\hat{\boldsymbol{P}}_i\}$ of the patches prior to triangulation by solving the following quadratic optimization problem:

$$\min_{\{\hat{\boldsymbol{P}}_i\}} \sum_{(i,j)\sim(k,l)} \|(\hat{\boldsymbol{P}}_i - \hat{\boldsymbol{P}}_j) + \boldsymbol{R}_S(\hat{\boldsymbol{P}}_k - \hat{\boldsymbol{P}}_l)\|_2^2 + \epsilon \sum_i \|\hat{\boldsymbol{P}}_i - \boldsymbol{P}_i\|_2^2. \tag{6.36}$$

Here, $\boldsymbol{R}_S = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$ represents the flip matrix, assuming the symmetry axis is vertical. This optimization involves solving a fixed-coefficient, positive definite linear system, which ensures differentiability. The *validated* edge control points $\{\hat{\boldsymbol{K}}^e\}$ are computed analytically by symmetrizing their relative coordinates. The symmetrization constraints are illustrated in Figure 6.37. Throughout this paper, we omit the hat notation for validated vertices and control points, as all computations are based on the symmetrized patches. However, it is important to note that the underlying garment optimization variables remain the original, non-symmetry-enforced geometry parameters.

Sewing Pattern Discretization To enable direct optimization of Bézier curves, we make the sampling from boundary curve parameters to mesh vertices differentiable. Both boundary sampling and interior sampling are conceptualized as fixed-coordinate sampling based on their control points. Each boundary edge curve $e \in \mathcal{E}$ is defined by the starting vertex $\boldsymbol{P}_0^e$, the control point $\boldsymbol{K}^e$, and the endpoint $\boldsymbol{P}_1^e$ (which is also the starting point of the next edge). The curve can be differentiably parameterized as $\boldsymbol{P}^e(t) = (1-t)^2\boldsymbol{P}_0^e + 2(1-t)t\boldsymbol{K}^e + t^2\boldsymbol{P}_1^e$. Uniform sampling along the curve in terms of arc length is represented as a set of parameters $\{t_1^e, \ldots, t_{n_e}^e\}$, with $\boldsymbol{V}_i^e = \boldsymbol{P}^e(t_i^e)$ being the sampled points. The number of sampled points n_e may vary for different edges. After independent triangulation for each patch, we compute the harmonic coordinate matrix $\boldsymbol{H} \in \mathbb{R}^{n_I \times n_B}$ (Joshi et al., 2007) for all the interior points, where n_I is the number of interior vertices and n_B is the total number of boundary vertices. With

a slight abuse of notation, we reparameterize the j-th interior vertex as $\boldsymbol{V}_j^I = \sum_i \boldsymbol{H}_{ji} \boldsymbol{V}_i^B$, with $\boldsymbol{H}_{ji}$ denoting its harmonic weight relative to the i-th boundary point $\boldsymbol{V}_i^B$. Here $\boldsymbol{H}_{ji}$ is zero if $\boldsymbol{V}_j^I$ and $\boldsymbol{V}_i^B$ do not belong to the same patch. During backpropagation, we fix the boundary sampling coordinates $\bigcup_{e \in \mathcal{E}, i \leq n_e} \{t_i^e\}$ and the interior harmonic coordinate matrix $\boldsymbol{H}$, so that the triangulation is analytically determined by the original parameters of the Bézier curves. These coordinates are updated only after remeshing is performed, which will be discussed in the garment optimization section.

6.4.5.2 Multi-view Image Generation

Given a single-view image of a full-body clothed human, we generate a set of multi-view RGB images and normal maps under orbital camera views using a pre-trained multi-view diffusion model, MagicMan (He et al., 2024b). These multi-view images of the clothed human are treated as ground truth data for human pose and garment shape in the subsequent reconstruction steps.

6.4.5.3 Human Body Reconstruction

The generated garment is statically draped on a fixed human mesh. To reduce the gap between the reconstructed garment and the image, an accurate human body is required to correctly support the garment. We use SMPL-X (Pavlakos et al., 2019) as our parameterized human model. First, we apply OSX (Lin et al., 2023b) to the input single-view image to obtain an initial pose estimation $\boldsymbol{\theta}$ and shape estimation $\boldsymbol{\beta}$. This initial estimation typically does not perfectly align with other views, and the scaling and rotation are inconsistent across the multi-view images. Subsequently, we fine-tune the pose based on multi-view images using a coarse-to-fine strategy.

In the coarse stage, we estimate joint landmarks on the images using DWPose (Yang et al., 2023c). Here, we optimize only the global scaling S and rotation $\boldsymbol{R}$ of the SMPL-X

model based on the following landmark loss:

$$\mathcal{L}^{\mathrm{P}}_{\mathrm{Land}} = \frac{1}{|\Omega|}\sum_{i} \|\boldsymbol{w}_i \cdot \left(\mathrm{Proj}(\boldsymbol{J}(S, \boldsymbol{R}, \boldsymbol{\theta}, \boldsymbol{\beta}); \Omega_i) - \bar{\boldsymbol{J}}_i\right)\|_2^2, \tag{6.37}$$

where $\Omega = \{\Omega_i\}$ represents the set of camera parameters, $\boldsymbol{J}$ is the 3D joint location map provided by the SMPL-X model, Proj is the projection operator from world space to screen space, $\bar{\boldsymbol{J}}_i$ is the 2D joint location estimated by DWPose, and $\boldsymbol{w}_i$ is the per-landmark confidence score of the estimation. We use $\|\cdot\|_2^2$ to denote the mean square error (MSE). This optimization essentially estimates the model-to-world matrix of the SMPL-X model. To further refine pose and shape parameters, in the fine stage, we additionally incorporate the following RGB loss and mask loss:

$$\mathcal{L}^{\mathrm{P}}_{\mathrm{RGB}} = \frac{1}{|\Omega|}\sum_{i} \|(\boldsymbol{M}_i^o)^c \cdot (\boldsymbol{I}(S, \boldsymbol{R}, \boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{C}_H; \Omega_i) - \bar{\boldsymbol{I}}_i)\|_1, \tag{6.38}$$

$$\mathcal{L}^{\mathrm{P}}_{\mathrm{Mask}} = \frac{1}{|\Omega|}\sum_{i} \|(\boldsymbol{M}_i^o)^c \cdot (\boldsymbol{M}(S, \boldsymbol{R}, \boldsymbol{\theta}, \boldsymbol{\beta}; \Omega_i) - \bar{\boldsymbol{M}}_i)\|_1. \tag{6.39}$$

Here, $\boldsymbol{C}_H$ represents the optimizable human body vertex color, while $\boldsymbol{I}(\cdot)$ and $\boldsymbol{M}(\cdot)$ denote the posed human body RGB rendering process and contour rendering process under camera view Ω_i, implemented using Nvdiffrast (Laine et al., 2020). $\bar{\boldsymbol{I}}_i$ and $\bar{\boldsymbol{M}}_i$ are the generated multi-view RGB images and masks, respectively. $\boldsymbol{M}_i^o$ represents the occluded region of the human body, which includes the garment region $\boldsymbol{M}_i^\beta$ and other non-garment occlusions $\boldsymbol{M}_i^\alpha$ (such as footwear, accessories, and hair). These regions are generated using SegFormer (Xie et al., 2021). The notation $(\cdot)^c$ denotes the complement of the specified region. We use $\|\cdot\|_1$ to denote the mean absolute error (MAE). By excluding the loss computation in the occluded region, we can accommodate loosely fitted garments. In summary, we optimize using the following loss in the fine stage:

$$\mathcal{L}^{\mathrm{P}}(S, \boldsymbol{R}, \boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{C}_H) = \mathcal{L}^{\mathrm{P}}_{\mathrm{RGB}} + \mathcal{L}^{\mathrm{P}}_{\mathrm{Mask}} + \lambda_1 \mathcal{L}^{\mathrm{P}}_{\mathrm{Land}} + \lambda_2\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|_1 + \lambda_3\|\boldsymbol{\beta} - \boldsymbol{\beta}_0\|_1. \tag{6.40}$$

Here, we also regularize the pose and shape parameters where $\boldsymbol{\theta}_0$ and $\boldsymbol{\beta}_0$ are their initial estimates provided by OSX.

6.4.5.4 Garment Initialization

The generated sewing patterns are positioned near the human body and sewn together to be dressed. SewFormer provides an initial placement around the T-posed SMPL-X model. To ensure proper layering, we adopt a bottom-to-top strategy for fitting the entire set of garments onto the human body, allowing the top garments to overlay the bottom ones. Connected components are identified by treating stitched vertices as connected. These components are sorted vertically and sequentially fitted from bottom to top through simulations using CIPC. After completing the T-pose fitting, the human body is interpolated from the T-pose to the reconstructed pose, and the entire cloth-human interaction is simulated by treating the human in motion as a moving boundary condition. To secure the bottom garments and prevent them from slipping during pose interpolation, we shrink the rest shape of the triangles near the waist to generate sufficient friction.

6.4.6 Garment Optimization

6.4.6.1 Optimization Overview

In garment optimization phase, we iteratively fine tune parameters of sewing pattern so that the statically draped garments on a posed human body match generated multi-view images in all views. We optimize the curve vertex set $\mathcal{P}$ and the control point set $\mathcal{K}$ of Bézier curves using differentiable CIPC simulation based on the generated multi-view images. To further leverage RGB information for assisting the optimization, we also optimize the vertex colors $\boldsymbol{C}_G$ of the discretized garment mesh for RGB renderings. Additionally, we optimize the global stretching stiffness κ_s and the global bending stiffness κ_b to automatically discover a set of physical parameters that align with the 2D observations.

For each optimization iteration, we use CIPC simulation to statically drape the garment onto the fixed-posed human body mesh. Leveraging the robustness of CIPC, we simulate one step of 1 second to directly reach near-static equilibrium. Since the static equilibrium does not locally depend on the initial state, meaning that the Jacobian matrix of the simulated

state with respect to the initial state is zero, we update the initial state of iteration n, $\boldsymbol{x}_0^n$, to the previously simulated state:

$$\boldsymbol{x}_0^n = \mathrm{Sim}(\boldsymbol{x}_0^{n-1}; \boldsymbol{\varsigma}(\kappa_s, \kappa_b, \mathcal{P}, \mathcal{K})). \tag{6.41}$$

Here, Sim represents the simulation process described in Section 6.4.3. The initial state, $\boldsymbol{x}_0^0$, is obtained from the initial garment fitting described in Section 6.4.5.4. $\boldsymbol{\varsigma}(\mathcal{P}, \mathcal{K})$ denotes the simulation rest shape data, including nodal mass, per-stencil elastic stiffness, undistorted material space, and similar properties. To make the simulation as path-independent as possible, we avoid adding friction during the process. To prevent the bottom garments from slipping down, the boundary loop of the bottom component near the waist area is fixed.

In summary, we solve the following optimization problem:

$$\min \mathcal{L}(\mathcal{P}, \mathcal{K}, \kappa_s, \kappa_b; \boldsymbol{x}, \boldsymbol{x}_0), \tag{6.42}$$

where $\boldsymbol{x}$ represents the simulated state starting from initial state $\boldsymbol{x}_0$, which is iteratively updated to the previously simulated state. We elaborate on the training losses in $\mathcal{L}$ that we use in the following sections. We observe that edge curvatures $\mathcal{K}$ are more sensitive than vertex positions $\mathcal{P}$. Therefore, we employ a two-stage training approach, where in the first stage, the update of $\mathcal{K}$ is frozen.

6.4.6.2 Rendering Losses

Garment Mask Loss The dominant rendering loss we employ is the garment mask loss. Given the multi-view ground-truth images, we use SegFormer (Xie et al., 2021) to segment top, bottom, and dress garment masks, assigning each component with a distinct color. The mask loss is defined as follows:

$$\mathcal{L}_{\mathrm{Mask}} = \frac{1}{|\Omega|} \sum_i \|(\boldsymbol{M}_i^{\alpha})^c \cdot (\boldsymbol{M}(\boldsymbol{x}; \boldsymbol{C}_C, \Omega_i) - \bar{\boldsymbol{M}}_i)\|_1. \tag{6.43}$$

Here, $\boldsymbol{x}$ represents the simulated state of garments draped over the human body. $\boldsymbol{C}_C$ denotes the component color, which is discussed in the following section. The rendered colored mask $\boldsymbol{M}(\boldsymbol{x}; \boldsymbol{C}_C, \Omega_i)$ is obtained by assigning $\boldsymbol{C}_C$ to the corresponding garment vertices and setting the human body to black, ensuring that only the non-occluded parts of the garments are rendered. $\bar{\boldsymbol{M}}_i$ is the set of colored garment masks generated from multi-view RGB images. We also exclude the loss computation in the occluded regions $\boldsymbol{M}_i^{\alpha}$ caused by hair and accessories to avoid incorrect mask guidance.

Remark 6.4.1. **Initialization of Component Colors.** The component color $\boldsymbol{C}_C$ is automatically assigned prior to garment optimization. SewFormer typically predicts garments with one or two connected components. We vertically sort the sewn garment components and the 2D mask regions from the first camera view. The component colors are then assigned accordingly. If only one component is predicted but multiple garment masks are present, we adjust the multi-view garment masks to use a single color.

RGB and Normal Rendering Loss We also utilize RGB and normal rendering losses to improve garment optimization. These losses are introduced to stabilize the training process, as the gradient of the mask rendering loss within the interior regions of the garment is zero. They are formulated similarly to the mask rendering loss:

$$\mathcal{L}_{\text{RGB}} = \frac{1}{|\Omega|} \sum_i \| \boldsymbol{M}_i^{\beta} \cdot (\boldsymbol{I}(\boldsymbol{x}; \boldsymbol{C}_G, \Omega_i) - \bar{\boldsymbol{I}}_i) \|_1, \tag{6.44}$$

$$\mathcal{L}_{\text{Normal}} = \frac{1}{|\Omega|} \sum_i \| \boldsymbol{M}_i^{\beta} \cdot (\boldsymbol{N}(\boldsymbol{x}; \Omega_i) - \bar{\boldsymbol{N}}_i) \|_1. \tag{6.45}$$

Here, $\boldsymbol{I}(\boldsymbol{x}; \boldsymbol{C}_G, \Omega_i)$ represents the garment RGB rendering of the vertex color $\boldsymbol{C}_G$ under the camera view Ω_i, and $\boldsymbol{N}(\boldsymbol{x}; \Omega_i)$ denotes the corresponding normal map rendering. The sets $\{\bar{\boldsymbol{I}}_i\}$ and $\{\bar{\boldsymbol{N}}_i\}$ are the multi-view RGB and normal images generated by the multi-view diffusion process. The loss computation is restricted to the garment regions $\boldsymbol{M}_i^{\beta}$.

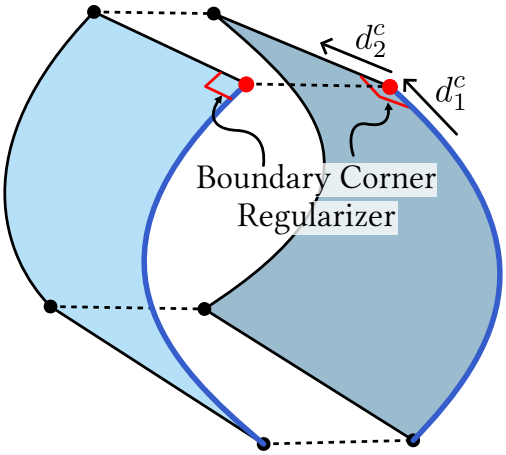


Figure 6.38: Illustration of boundary corner regularizer.

6.4.6.3 Geometric Regularizers

The sewing pattern optimization under rendering losses alone is ill-posed because, for the same sewn 3D garment mesh, there are infinitely many ways to decompose the mesh into flattened patches. Therefore, we incorporate several geometric losses to regularize the sewing pattern optimization.

6.4.6.4 Area Ratio Loss

We use the following area ratio loss to preserve the relative area of each patch with respect to the connected component it belongs to:

$$\mathcal{L}_{\text{AR}} = \frac{1}{N_P} \sum_p \left(\frac{\bar{A}_p(\boldsymbol{X})}{\bar{A}_p(\boldsymbol{X}^0)} - 1 \right)^2, \tag{6.46}$$

where N_P is the number of garment patches, $\bar{A}_p$ is the operator that computes the ratio between the area of the p-th patch and the area of the component. $\boldsymbol{X}$ represents the current 2D discretization of the garment patches, and $\boldsymbol{X}^0$ denotes the initial sampling.

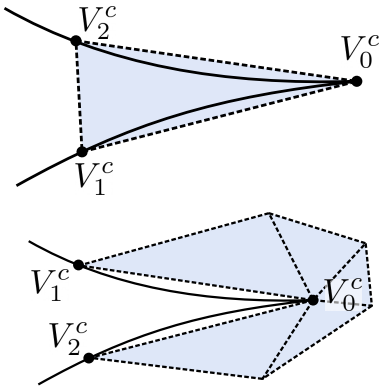


Figure 6.39: Illustration of small angles where the regularization is applied.

6.4.6.5 Corner Regularizers

Boundary Corner Regularizer For the boundary loops of garment components, we identify all corner vertices of the original Bézier curves. At these corners, where two patches are typically sewn together, we apply the following boundary corner regularizer to penalize deviations of corner angles from right angles, as illustrated in Figure 6.38:

$$\mathcal{L}_{\text{BC}} = \frac{1}{N_{BC}} \sum_{c} (1 - \boldsymbol{d}_1^c \times \boldsymbol{d}_2^c). \tag{6.47}$$

Here, N_{BC} represents the total number of boundary corners, $\boldsymbol{d}_1^c$ and $\boldsymbol{d}_2^c$ denote two consecutive unit tangent vectors at corner c.

Small-Angle Corner Regularizers Small angles at patch corners can introduce instabilities into optimization and simulation; thus, we use the following regularizer to penalize such angles:

$$\mathcal{L}_{\text{SAC}} = -\frac{1}{N_C} \sum_{c} s_c(\boldsymbol{X}) (\widehat{\boldsymbol{V}_1^c - \boldsymbol{V}_0^c}) \times (\widehat{\boldsymbol{V}_2^c - \boldsymbol{V}_0^c}). \tag{6.48}$$

Here, N_C is the number of patch corners, $(\boldsymbol{V}_1^c, \boldsymbol{V}_0^c, \boldsymbol{V}_2^c)$ is the tuple of three consecutive discrete boundary sampling points at the corner c, $\widehat{(\cdot)}$ is the vector normalization operator. $s_c(\boldsymbol{X})$ is a non-differentiable sign function: $s_c(\boldsymbol{X}) = 0$ if the discretized corner angle is larger than some threshold, otherwise, $s_c(\boldsymbol{X})$ equals the sign of the cross product on the initial sewing pattern. This regularization applies to the two cases illustrated in Figure 6.39. It tries

to maintain the same sign of the angle and avoid the angle becoming too small. However, Bézier curves may still intersect at corners even though the discretized corner triangles are normal. We use the following discretization consistency regularizer to align the curve's end tangents and discrete edge directions:

$$\mathcal{L}_{\text{DC}} = \frac{1}{N_C} \sum_c (2 - \boldsymbol{\tau}_1^c \cdot (\widehat{\boldsymbol{V}_1^c - \boldsymbol{V}_0^c}) - \boldsymbol{\tau}_2^c \cdot (\widehat{\boldsymbol{V}_2^c - \boldsymbol{V}_0^c})), \tag{6.49}$$

where $\boldsymbol{\tau}_1^c, \boldsymbol{\tau}_2^c$ are two consecutive end tangents of Bézier curves at corner c.

6.4.6.6 Comfort Loss

In addition to the appearance of the fitting matching the observation, we also aim to ensure that the fitting is comfortable. We use the stretching elasticity energy to evaluate the tightness of the fitting. To prevent overly tight fitting, we introduce the following comfort regularizer:

$$\mathcal{L}_{\text{Comfort}} = \int \|\boldsymbol{F}(\boldsymbol{x}, \boldsymbol{X}) - \boldsymbol{R}(\boldsymbol{F})\|^2 d\boldsymbol{X}, \tag{6.50}$$

where $\boldsymbol{R}(\boldsymbol{F})$ represents the closest rotation matrix to $\boldsymbol{F}$. This is the same as the as-rigid-as-possible (ARAP) stretching energy used in the forward simulation, except that here we assume the global stiffness is 1.

6.4.6.7 Laplacian Loss

To ensure the smoothness of the fitting, we include a Laplacian regularizer:

$$\mathcal{L}_{\text{Lap}} = \|\Delta \boldsymbol{x}\|_2, \tag{6.51}$$

where Δ represents the node-area-weighted Laplacian operator on triangle meshes, and $\boldsymbol{x}$ denotes the simulated garment vertex positions.

6.4.6.8 Seam Losses

The stitched curved edge pairs should have the same shape to prevent undesired wrinkles near the seams. To achieve this, we use a seam length regularization similar to (Li et al., 2024d) to regularize the paired stitched edges:

$$\mathcal{L}_{\text{SL}} = \frac{1}{N_S} \sum_{e_i \sim e_j} \left| \int \|\dot{\boldsymbol{P}}^{e_i}(t)\| dt - \int \|\dot{\boldsymbol{P}}^{e_j}(t)\| dt \right|, \tag{6.52}$$

where N_S is the number of stitched seams, e_i and e_j iterate over all stitched edge pairs, and $\dot{\boldsymbol{P}}^{e}(t)$ represents the tangent vector. The integral is computed using finite difference and the Riemann sum. Additionally, we regularize the seam curvatures on these pairs to preserve their initial curvatures:

$$\mathcal{L}_{\text{SC}} = \frac{1}{2N_S} \sum_{e_i \sim e_j} \|\bar{\mathcal{K}}^{e_i} - \bar{\mathcal{K}}^{e_i,0}\| + \|\bar{\mathcal{K}}^{e_j} - \bar{\mathcal{K}}^{e_j,0}\|, \tag{6.53}$$

where $\bar{\mathcal{K}}^{e}$ represents the relative coordinate of the control point within the frame of the curved edge segment e, and $\bar{\mathcal{K}}^{e,0}$ denotes its initial value.

6.4.6.9 Post-Iteration Processing

Occasionally, when two Bézier curves come close to each other—such as when forming a thin strip—the curves may penetrate one another after a parameter update in some iteration. This can lead to flipped triangles, causing the simulation to fail in the next iteration. To address this, we enforce a safeguard that modifies the geometry in-place to prevent such occurrences. Specifically, we optimize the negative triangle areas using a least-squares penalty after each iteration n:

$$\mathcal{L}^{\text{Flip}}(\mathcal{P}, \mathcal{K}) = \frac{1}{|F|} \sum_{f} (\epsilon - \min\{A_f(\boldsymbol{X}), \epsilon\}) + \lambda^{\text{Flip}} \|\boldsymbol{X} - \boldsymbol{X}^{n+1}\|_1, \tag{6.54}$$

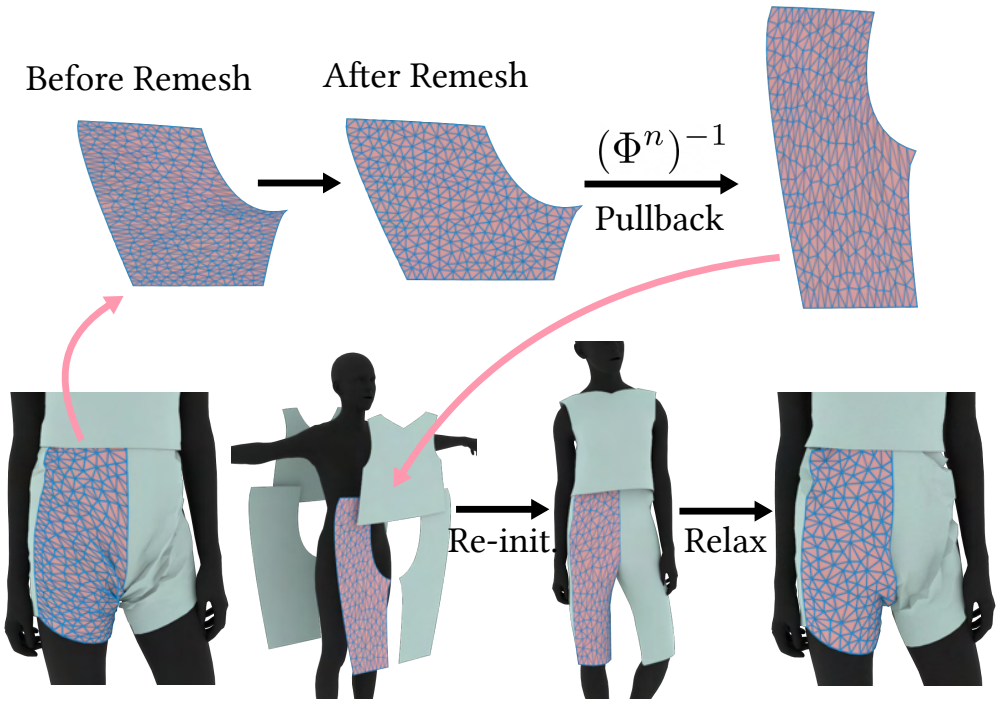


Figure 6.40: **Sewing Pattern Remeshing.** We perform automatic remeshing during optimization when ill-conditioned triangles are detected. To avoid penetration, we pull back the new discretization to the initial unoptimized stage and rerun the garment initialization to fit it onto the human.

where $|F|$ is the number of faces A_f is the signed area of triangle f and $\boldsymbol{X}^{n+1}$ is the discretized garment vertices after the parameter update at iteration n. We optimize the above loss only if no triangles are close to flipping.

6.4.6.10 Remeshing

During optimization, we use cage deformations defined by a fixed set of harmonic coordinates to deform a fixed number of interior vertices. The triangulation quality can degrade significantly in regions with large deformations, creating challenges for simulations. To address this, we introduce automatic remeshing during the optimization iterations when the mesh quality drops below a predefined threshold. While rerunning the discretization on updated sewing patterns is straightforward, directly remeshing the fitted garment state on the human body can lead to penetrations. This occurs because the underlying smoothly interpolated surface may

intersect after re-triangulation, as the collision handling relies on the previous discretization. To resolve this, we propose a refitting procedure that sews and refits the garment patches onto the human body without causing penetrations.

Assume $\boldsymbol{\chi}^0$ is the initial garment sewing pattern in the continuous domain with triangulation $\mathcal{T}^0$. The sewing pattern optimization at step n can be characterized by a map Φ^n from $\boldsymbol{\chi}^0$ to $\boldsymbol{\chi}^n$, where Φ^n is a piecewise linear map defined on the continuous domain. Observe that the initial fitting is sewn from the discretization of $\boldsymbol{\chi}^0$, where SewFormer provides reasonable transformations to position the panels around the human body. After generating a new triangulation $\mathcal{T}^n$ of $\boldsymbol{\chi}^n$, we pull $\mathcal{T}^n$ back to $\boldsymbol{\chi}^0$ as the new triangulation of $\boldsymbol{\chi}^0$: $\tilde{\mathcal{T}}^0 \leftarrow [\Phi^n]^{-1}(\mathcal{T}^n)$. We then apply the initial transformations to the updated discretization $\tilde{\mathcal{T}}^0$ to position the patches around the T-pose human body and execute the fitting procedure described in Section 6.4.5.4. During this fitting process, we set the rest shape as $\tilde{\mathcal{T}}^0$. A relaxation process follows, using $\mathcal{T}^n$ as the rest shape. The newly fitted results are non-penetrating, and we set them as the initial state $\boldsymbol{x}_0^n$ for the differentiable simulation process. Finally, $\mathcal{T}^0$ is replaced with $\tilde{\mathcal{T}}^0$. This remeshing process is illustrated in Figure 6.40.

6.4.7 Post-Optimization Steps

6.4.7.1 Texture Generation

To complete our pipeline and deliver a fully textured garment directly from a single image input, tailored to the needs of the garment fabrication industry, we incorporate an additional texture generation module. Unlike formulating texture creation as a reconstruction task—an approach constrained by the ill-posed nature of the problem due to sparse inputs, severe distortion, and occlusions caused by the human body and overlapping garment layers—our module adopts generative methods to produce garment textures. This module employs two strategies for texture generation:

Tileable Texture Generation via FabricDiffusion In this strategy, we assume that in real-world garment creation, clothing panels are typically cut from a single piece of fabric and

sewn together, resulting in similar and tileable textures. Based on this assumption, given the front-view ground truth input image and its corresponding colored segmentation mask, we identify the largest uniform color square area within the segmentation mask for each garment component (e.g., top or bottom) as the captured texture region. This region may exhibit distortions and varying illumination caused by occlusions and poses in the input image. To address these issues, we process the captured texture region using FabricDiffusion (Zhang et al., 2024), which generates distortion-free and tileable texture maps. To determine the appropriate tiling scale for aligning the textures with the garment's UV space (optimized in our pipeline), we assume consistent camera view parameters for the front view. This scale can be calculated by multiplying the derivative of the cropped region's size by a constant factor.

In-the-Wild Texture Generation via GPT-4o and FLUX For generalized textures that do not fall into the above case, we utilize Vision-Language Models (VLMs) in collaboration with a Diffusion model. Specifically, we process the input image using the GPT-4o (Achiam et al., 2023) VLM to extract descriptive keywords for the textures of various components, such as `"denim, dark blue, smooth fabric"` and `"argyle, grey and white, knitted fabric"` through prompt-based querying. These extracted keywords are then fed into FLUX (Labs, 2023), which generates the corresponding textures.

6.4.7.2 Showcase under Human Motions

The reconstructed simulation-ready garment and human model can be used to generate realistic dynamic human motion in clothing using the robust CIPC physics-based simulator. However, IPC-based simulators require the initial configuration of the human model to be penetration-free. As IPC-based simulators produce intersection-free results, self-penetrations of the human model during the given motion can cause solution failures when the human model interacts with garments. To address this issue, we replace the human model during motion with the nearest intersection-free human model by solving Injective Deformation Processing (IDP) (Fang et al., 2021) problems. In solving these IDP problems, we follow

the method in Li et al. (2024a) where the authors use PBD simulations to resolve collisions between garments and human models. Instead, we apply extended Position-Based Dynamics (XPBD) (Macklin et al., 2016) simulations to resolve self-penetrations in the human model by repelling colliding vertices and faces, while preserving natural deformations.

6.4.8 Implementation

Differentiable Simulation Layer We implement CIPC simulation using NVIDIA Warp (Macklin, 2022) to utilize the Auto-Diff feature. The simulator is wraped in a customized `autograd.Function` to be integrated to the global computational graph.

Balancing between Losses The training loss is weighted sum of all rendering losses and geometric regularizers. We use the L_{Mask} as the dominant loss and set the relative weights for L_{RGB} and L_{Normal} to $\lambda_{\text{RGB}} = 0.1$ and $\lambda_{\text{Normal}} = 0.1$. For geometric regularizers, we do not have a rule of thumb to balance them. For all experiments, we use $\lambda_{\text{Lap}} = 0.001$, $\lambda_{\text{Comfort}} = 0.1$, $\lambda_{\text{AR}} = 0.01$, $\lambda_{\text{SAC}} = 0.01$, $\lambda_{\text{DC}} = 0.001$, $\lambda_{\text{BC}} = 0.001$, $\lambda_{\text{SL}} = 0.1$, $\lambda_{\text{SL}} = 0.1$.

Training Time The pre-optimization steps requires about 10 minutes. The garment optimization process can be finished within 2 hours on a single RTX 3090 with 24GB device memory.

6.4.9 Experiments

6.4.9.1 Geometry Reconstruction Comparison

We first conduct comparison study to evaluate the reconstruction accuracy of baseline methods and our proposed approach.

Benchmark We use the CloSe (Antić et al., 2024) and 4D-Dress (Wang et al., 2024) datasets for the comparison study. CloSe is a large-scale 3D clothing dataset featuring detailed segmentation across diverse clothing classes. 4D-Dress offers high-quality 4D textured

Table 6.7: **Quantitative Comparisons of Geometry Reconstruction.** We evaluate the performance of baseline methods and our approach on the CloSe and 4D-Dress datasets. Our proposed method achieves the highest reconstruction accuracy across both datasets.

Method	CloSe		4D-Dress	
	CD↓	IoU↑	CD↓	IoU↑
BCNet	2.277	0.781	4.704	0.575
ClothWild	2.166	0.664	3.125	0.664
GarmentRecovery	2.058	0.831	2.983	0.776
SewFormer	2.233	0.748	2.926	0.720
Dress-1-to-3 (Ours)	**1.623**	**0.862**	**2.441**	**0.808**

scans of dynamic clothed human sequences. For evaluation, we carefully select examples encompassing a variety of human body shapes, poses, and their corresponding front-view images to establish a comprehensive benchmark.

Baselines pare our method with state-of-the-art single-view garment reconstruction methods, including BCNet (Jiang et al., 2020a), ClothWild (Moon et al., 2022), GarmentRecovery (Li et al., 2024c), and SewFormer (Liu et al., 2023a). Among these, BCNet and ClothWild are designed for clothed human reconstruction but are limited to tight-fitting clothing and not readily adaptable for downstream tasks such as animation and simulation. GarmentRecovery extends to loose-fitting garments reconstruction by deforming predicted rest shapes to align with input images. In contrast, SewFormer predicts corresponding sewing patterns directly from images, enabling seamless integration into animation pipelines and physical simulations. Our proposed method builds upon SewFormer and incorporate differentiable simulation to refine 2D panels and physical parameters. For SewFormer and our approach, we simulate the predicted sewing patterns and use the resulting 3D garments for quantitative comparisons.

Results We evaluate the accuracy of baseline methods and our approach using two metrics: Chamfer Distance (CD) and Intersection over Union (IoU). CD quantifies the geometric similarity between reconstructed and ground-truth meshes, while IoU assesses the alignment between the garment mask of the rendered reconstruction and the input front-view images. The quantitative results for the CloSe and 4D-Dress datasets are presented in Table 6.7, and

Figure 6.41: **Qualitative Comparisons of Geometry Reconstruction.** Our proposed method not only generates sewing patterns that seamlessly integrate into animation and simulation workflows but also achieves superior garment reconstruction accuracy compared to baseline methods.

visualized qualitative comparisons are shown in Figure 6.41. BCNet and ClothWild tend to produce overly smooth garment meshes, lacking fine wrinkle details. GarmentRecovery improves geometric details but often results in interpenetrated reconstructions. SewFormer predicts sewing patterns that can be directly used for simulation, yet it neglects physical parameters, leading to simulated results that deviate significantly from the ground-truth

mesh. In contrast, our method not only generates sewing patterns for seamless integration into downstream pipelines but also optimizes garment physical parameters, enabling accurate geometry reconstruction that closely aligns with ground truth.

6.4.9.2 Sewing Pattern Evaluation

We compare our method with two approaches that predict garment sewing patterns: Neural Tailor (Korosteleva and Lee, 2022) and SewFormer (Liu et al., 2023a). Neural Tailor generates sewing patterns from garment point cloud inputs, while SewFormer and our method recover patterns directly from single-view image inputs. For comparison purposes, we sample points from garment meshes to serve as inputs for Neural Tailor. The qualitative results are shown in Figure 6.42. Neural Tailor is trained on garments draped over an average SMPL female body in a T-pose. Consequently, its predictions are usually unsatisfactory if the human pose deviates from the T-pose, and may produce unexpected additional panels. Furthermore, its reliance on garment point clouds as input significantly restricts its practical applicability. SewFormer, on the other hand, generates symmetric and organized panels. However, its predicted panel shapes often fail to align with the input image. For instance, it may predict long pant panels for an image with short pants. This is probably due to the small scale of the dataset used for its training. Nevertheless, collecting a large-scale dataset of real-world clothed human images paired with corresponding garment meshes and sewing patterns is a challenging and resource-intensive task. In contrast, our optimization-based method requires no additional training data. By leveraging differentiable simulation, it refines an initial estimate of the sewing patterns, achieving significantly more accurate results.

6.4.9.3 Textured Garment Reconstruction and Simulation

Test Images To evaluate the generative capability of our method, we perform extensive tests on a variety of images from different sources. We begin by utilizing real-world images from the DeepFashion2 (Ge et al., 2019) dataset, which comprises in-the-wild clothing images sourced from the internet, including those from commercial shops and customers.

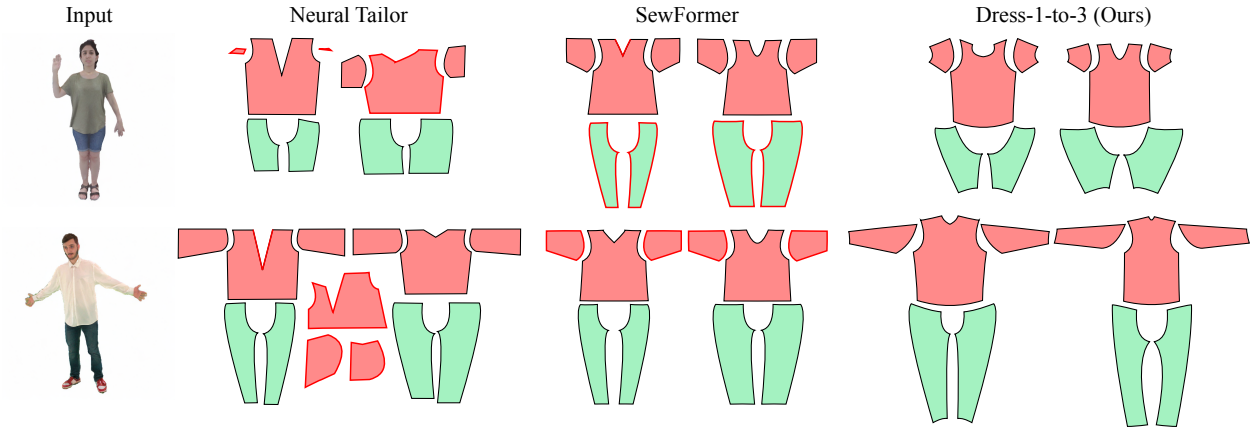


Figure 6.42: **Qualitative Comparison of Panel Shape Prediction.** Neural Tailor (Korosteleva and Lee, 2022) takes ground-truth garment meshes as input, while SewFormer (Liu et al., 2023a) and our proposed method use single-view images as input. Extra unexpected panels and edges with significant errors are highlighted in red.

These images exhibit varying quality and are captured under arbitrary camera and scene settings, reflecting casual user captures. In addition to real-world images, we evaluate our model using synthetic fashion images from the DeepFashion-Multimodal (Jiang et al., 2022) dataset. This dataset features high-resolution clothing images of various types, generated by a diffusion-based transformer. To further extend the generative capability of our method with text prompts, we employ FLUX (Labs, 2023) to generate input images using custom textual descriptions of clothing on a model. For instance, prompts such as `"a female model wearing a blazer and pants"` are used. To enhance the diversity of the generated results, we randomly incorporate detailed descriptions, including the shape and color of the clothing, as well as the pose and appearance of the model.

Textured Garment Reconstruction As demonstrated in Figure 6.43, Dress-1-to-3 effectively reconstructs 3D garments that accurately fit human models in both real-world and synthetic images. Our method automatically retrieves visually plausible garment textures using image diffusion techniques. This streamlined process requires minimal human effort to reconstruct high-fidelity garments with sewing patterns and offers users the flexibility to easily adjust garment shape and texture.

Figure 6.43: **Qualitative Results of Textured Clothed Human.** We showcase the generation capability of Dress-1-to-3 using in-the-wild test images from various sources, including both real-world and synthetic images. Our streamlined pipeline generates perfectly fitted 3D garments with visually plausible textures.

Garment Simulation The garments synthesized by our method are simulation-ready due to the accurate sewing, fitting, and optimization of garment patterns. The optimized 3D outfits align perfectly with the human body at steady state, avoiding artifacts such as self- or interpenetration. These garments can be seamlessly integrated into physics-based simulations, such as those used in video games. In Figure 6.35 and Figure 6.44, we visualize several simulated human motion sequences showcasing dynamic garment behavior.

Figure 6.44: **Garment Simulation.** We animate garment motion using various human sequences as moving boundary conditions. Our simulation-ready garments exhibit physically plausible dynamics.

6.4.9.4 Ablation Study

In Figure 6.45, we perform an ablation study for key individual components in Dress-1-to-3, using the same garment images as in Section 6.4.9.3. This study evaluates the contributions of each proposed component to the final garment reconstruction quality.

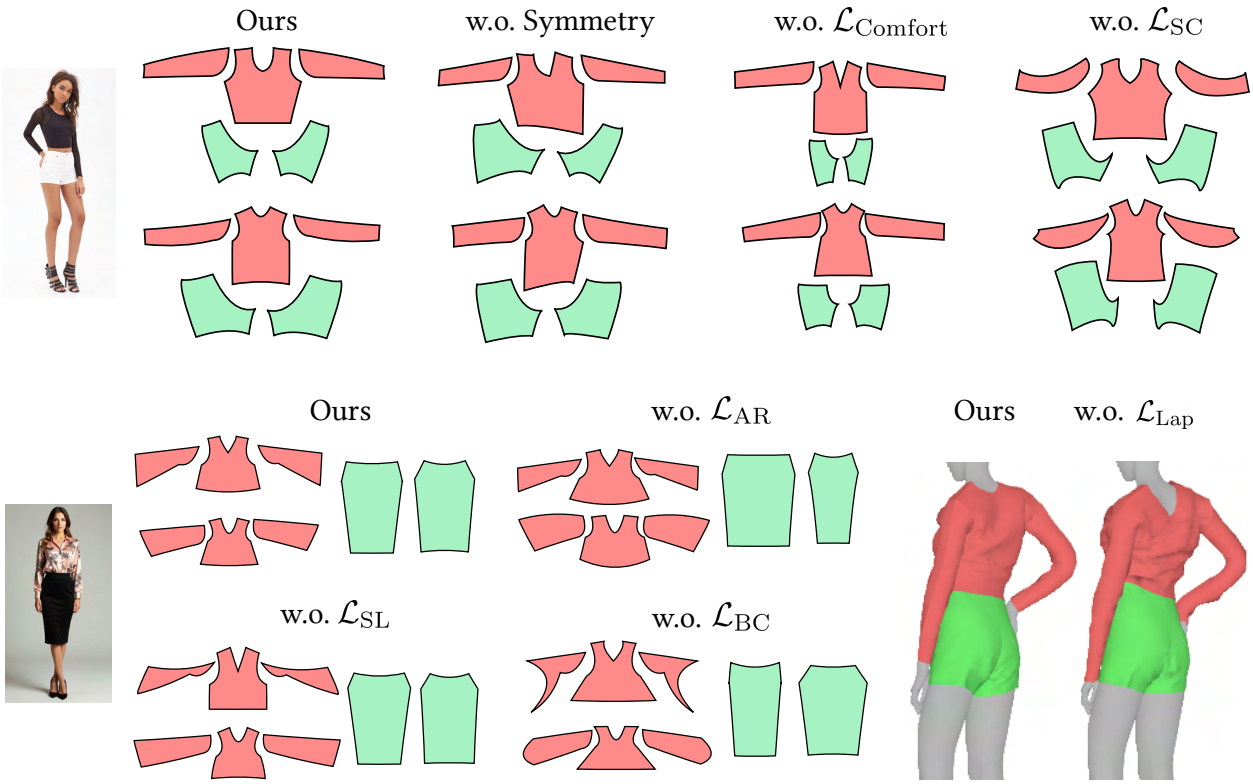


Figure 6.45: **Ablation Study.** We conduct ablation studies on our geometric regularizer to ensure that the sewing pattern maintains both reasonable 2D patterns and a plausible 3D fitted shape. We minimize irregularities such as asymmetry, sharp or acute angles, and inconsistent scaling of the 2D patterns while reducing noisy geometry and unrealistic wrinkles.

Patch Symmetrization We first evaluate the effectiveness of the proposed patch symmetrization, designed to facilitate better capture of symmetrical geometry. As shown in Figure 6.45, removing symmetry enforcement results in visibly asymmetric outputs compared to the input garment image. This highlights the critical role of symmetry enforcement in preserving structural coherence and alignment, particularly for garments with strong symmetrical patterns, such as dresses or jackets. By aligning the reconstructed mesh to expected symmetrical features, this component ensures geometric fidelity.

Laplacian Loss Laplacian loss $\mathcal{L}_{\text{Smooth}}$ is applied to smooth out noise and irregular wrinkles in the reconstructed garment mesh. This loss minimizes high-frequency artifacts, enabling a cleaner and more aesthetically pleasing surface. The weight of $\mathcal{L}_{\text{Smooth}}$ is a tunable parameter, allowing users to control the degree of smoothness based on their preferences. As shown

in Figure 6.45, a higher weight results in smoother results but may slightly reduce detail, whereas a lower weight preserves intricate wrinkles but may retain noise.

Boundary Corner Regularizer The boundary corner regularizer, $\mathcal{L}_{\text{BC}}$, mitigates the occurrence of sharp angles in the reconstructed sewing patterns. Sharp or acute angles can lead to practical difficulties during garment fabrication, as they introduce challenges in stitching and material handling. As demonstrated in Figure 6.45, models trained without $\mathcal{L}_{\text{BC}}$ often generate sewing patterns with acute or impractical corner geometries, whereas incorporating this regularizer results in smoother, more fabrication-friendly boundaries.

Comfort Loss Comfort loss, $\mathcal{L}_{\text{Comfort}}$, ensures the reconstructed garment mesh adheres to an appropriate scale relative to the input image. This prevents the generation of sewing patterns that are too small or tight, which could compromise wearability. Without $\mathcal{L}_{\text{Comfort}}$, as shown in Figure 6.45, the reconstructed sewing patterns often exhibit significantly smaller dimensions than expected, leading to impractical or unrealistic results. Incorporating this loss ensures that the final garment size aligns with user expectations and real-world usability requirements

Area Ratio Loss To maintain realistic proportions between garment parts, the area ratio loss, $\mathcal{L}_{\text{AR}}$, is applied to ensure that the relative area of each patch remains consistent with the connected components, reflecting real-world fabrication principles. For instance, in a skirt, the front and back panels should have comparable areas to align with practical garment construction. As illustrated in Figure 6.45, omitting $\mathcal{L}_{\text{AR}}$ often results in disproportionate patch sizes, such as an overly large front skirt panel compared to the back, violating fabrication norms.

Seam Losses Two seam losses: the length seam loss, $\mathcal{L}_{\text{SL}}$, and the curvature seam loss, $\mathcal{L}_{\text{SC}}$ are adopted to ensure that stitched curved edge pairs should have the same shape to prevent undesired wrinkles near the seam, and that enforces preservation of seam curvatures, respectively. As shown in Figure 6.45, the absence of $\mathcal{L}_{\text{SL}}$ leads to uneven sleeve seams,

introducing visual artifacts and potential fabrication issues. Similarly, without $\mathcal{L}_{\text{SC}}$, the seams can become overly curved, deviating significantly from the intended design. Together, these losses contribute to producing smooth and realistic seams.

6.4.10 Conclusion

In this paper, we present a garment reconstruction pipeline, Dress-1-to-3, which takes a single-view image as input and reconstructs a posed human wearing textured garments, with both the human pose and garment shapes closely aligned with the input image. During optimization, we refine the sewing pattern shapes and physical material parameters by leveraging a differentiable CIPC simulator with accurate frictional contact. The resulting garment assets are simulation-ready and can be seamlessly integrated into a physics-based simulator.

We benchmark our pipeline against baseline methods through two key experiments: a quantitative comparison of geometry reconstruction using existing garment datasets and a qualitative evaluation of sewing patterns. In both cases, Dress-1-to-3 significantly outperforms the baseline approaches.

To further assess the Dress-1-to-3's robustness and performance, we test our textured garment reconstruction using in-the-wild real-world and synthetic images, validated together with animations of dressed humans. The high-quality results demonstrate the robustness and effectiveness of our approach.

Additionally, ablation studies underscore the importance of the patch symmetrization technique and the contributions of each regularization loss term, highlighting their critical role in optimizing the pipeline's performance.

Limitations and Future Work While our method provides consistent high-fidelity reconstruction and has been extensively tested with in-the-wild images, its generation ability is somewhat limited by the initial estimation of the sewing pattern. For instance, our method cannot predict new connected pattern components if they are not included in the initial

estimation. Additionally, challenges arise with layered clothing, as SewFormer can only predict single-layer patterns, causing multi-layered garments to be fused into a single cloth component. It is worth noting that with a more versatile sewing pattern predictor capable of handling such cases, our method would also be able to process more complex garments. We leave this enhancement as future work.

CHAPTER 7

Sim-to-Real Applications

7.1 Lagrangian-Eulerian Multi-Density Topology Optimization with the Material Point Method

7.1.1 Introduction

Topology optimization is experiencing a rapid advance over the past few years, thanks to the collision of waves between next-generation computing infrastructure and high-performance simulation software. A surge of recent work has been creating various computing infrastructures capable of accommodating topology optimization applications with a super-scale resolution—millions to one billion of material voxels—on parallelizable data structures (*e.g.*, (Aage et al., 2017; Liu et al., 2018; Wu et al., 2015)). These density-based approaches stem from the conventional Solid Isotropic Material with Penalization Method (SIMP) (Sigmund, 2001; Andreassen et al., 2011) and naturally fall into Eulerian methods, owing to their geometric representation of the material evolution on a fixed grid. Level set(Osher and Sethian, 1988)-based methods(Wang et al., 2003; Allaire et al., 2004; Luo et al., 2008; van Dijk et al., 2013) are also Eulerian due to an implicit representation of the topology on grid nodes. On the other hand, Lagrangian geometries are increasingly attracting attention. For example, particles (*e.g.*, in smoothed-particle hydrodynamics (SPH) (Gingold and Monaghan, 1977)) can explicitly track the structural evolution under the guidance of material derivatives. Tracking explicit meshes is also a promising direction thanks to the advent of high-performance meshing software (Christiansen et al., 2014).

7.1.1.1 Hybrid Representation

Despite extensive research, Eulerian approaches have limited capability in capturing intricate structures, especially when the problem requires fine features that are hierarchical, codimensional, and can emerge from a nihil. On the other hand, Lagrangian representations suffer from a lower computational performance. Analogous to their computational physics counterparts (*e.g.*, in computational fluid dynamics), Eulerian approaches are not naturally adaptive to subgrid features, whereas Lagrangian methods face challenges in establishing differential stencils that are geometrically symmetric and numerically accurate.

In computational physics, researchers face the same dilemma regarding the choice of data structures and the corresponding numerical stencils when simulating large-scale fluids and solids. This dilemma further triggered the invention of a bank of hybrid Lagrangian-Eulerian methods, such as Particle-In-Cell (PIC) / Fluid-Implicit-Particle (FLIP) methods (Harlow, 1962; Brackbill et al., 1988) and Material Point Methods (MPM) (Sulsky et al., 1995; de Vaucorbeil et al., 2019), which are featured by an Eulerian background grid as a scratch pad and a set of Lagrangian particles to track geometry and topology. By conducting data transfers between the two representations, a hybrid Lagrangian-Eulerian scheme can typically leverage both sides' merits, enabling flexible and robust numerical solutions (Brackbill et al., 1988; Sulsky et al., 1995; Zhang et al., 2016c).

Motivated by such a design philosophy, some hybrid methods are also proposed for topology optimization. For example, the Moving Morphable Component (MMC) method (Guo et al., 2014; Zhang et al., 2016b, 2017b,d,c; Lei et al., 2019) aims to substantially reduce the number of design variables by optimizing component-wise distributions. It represents structures by unions of superellipse level sets – low dimensional morphable components that can move, deforming, and overlapping to track topology changes. The explicit geometric information also helps control the minimum length scale (Zhang et al., 2016a). MMC can produce results with sharp features with attractive convergence and timing profiles. However, to acquire sophisticated geometry features, a large number of components, *i.e.*, design variables, is necessary. On the other hand, the Moving Node Approach (MNA) (Overvelde,

2012) represents the target shape with a set of mass nodes, of which the positions are optimized to search for an optimal structure. In MNA, the quadrature is a set of regularly sampled discretization nodes, on which the densities are computed according to the clustering of mass nodes. However, such an approach can lead to results with many isolated mass nodes disconnected from the main structure, which can only be cleaned up using an extra post-processing step.

Inspired by these hybrid methods, a new hybrid Lagrangian-Eulerian topology optimization method—LETO is proposed in this paper. This new approach optimizes material distributions over a design domain by evolving a set of material carrier particles on a background Cartesian grid. With MPM applied for solving the static equilibrium, another set of particles, each carrying a temporally varying density, is evolved as a Lagrangian representation of material distribution, eliminating redundant, isolated particle blobs. The Lagrangian-Eulerian nature of the framework enables the communication between the moving particles and the fixed background MPM quadrature points by transferring the density values through interpolation functions. More specifically, as the carrier particles move and change their densities, the quadrature points' density values are updated accordingly, naturally providing sub-cell density resolution. As shown in the experiments, LETO tends to generate structures with rich branching fibers with low compliance.

7.1.1.2 Topology Optimization with the Material Point Method

When applying the adjoint method on topology optimization for sensitivity analysis, elasticity simulation is required in each iteration to obtain the nodal displacements under force equilibrium given a material distribution and an external load. A static elasticity solver can be applied since inertia effects are often ignored. While traditional topology optimization methods often use a grid-based Finite Element discretization, we adopt MPM for the static setting with the sub-cell resolution achieved by assigning a different density value to each quadrature particle. The static force equilibrium is further solved with a variational formulation that guarantees robustness and stability.

Spatial Discretization Traditional topology optimization methods, including both density-based approaches (Sigmund, 2001; Buhl et al., 2000) and level set-based approaches (Guo et al., 2014; Wang et al., 2003), often apply grid-based Finite Element discretization for the static solve. In FEM, the same material density is assigned for all quadratures in every single cell. Therefore, the domain boundaries are formed with jagged finite element edges, and even plotting zero-level contour still results in jagged boundaries (Maute and Sigmund, 2013). A grid with higher resolution is often required to alleviate these artifacts, which increases the computational cost.

MPM is a hybrid Lagrangian-Eulerian method widely used in different fields, *e.g.*, computer graphics (Stomakhin et al., 2013; Wolper et al., 2019), civil engineering (Abe et al., 2014; Zabala and Alonso, 2011), mechanical engineering (Sulsky et al., 1995; Guilkey and Weiss, 2003; Sinaie et al., 2018; Chen and Brannon, 2002). With the capability of handling large deformation (Nair and Roy, 2012; Charlton et al., 2017; Nguyen and Nguyen, 2016; Sadeghirad et al., 2011; Soga et al., 2016; Lian et al., 2012), topology changes, and coupled materials, MPM has been considered as one of the top choices in various physics-based simulations, including fracture (Guo and Nairn, 2006; Wolper et al., 2019; Yang et al., 2014; Long et al., 2019, 2016; Homel and Herbold, 2017), viscoelastic and elastoplastic solids (Fang et al., 2019; Burghardt et al., 2012), incompressible materials (Kularathna and Soga, 2017; Zhang et al., 2017a), high explosive explosion (Ma et al., 2009), snow (Stomakhin et al., 2013; Gaume et al., 2018, 2019), granular material (Bardenhagen et al., 2000; Klár et al., 2016; Yerro et al., 2019; Zhang et al., 2009) and mixtures (Gao et al., 2018a; Tampubolon et al., 2017; Bandara and Soga, 2015). In MPM, Lagrangian particles, which are also known as material points, are used to track quantities like mass, momentum, and deformation. On the other hand, a regular Eulerian grid is built to evaluate force and update velocity at each time step. Particle quantities are then updated from the interpolation of nodal quantities. MPM's convergence was demonstrated computationally and explained theoretically with a smooth, e.g., quadratic B-spline, basis for grid solutions (Steffen et al., 2008), which was further verified with manufactured solutions (Wallstedt, 2009). MPM is applied as the spatial discretization in this work. We also describe a static formulation for directly solving the

force equilibrium, which allows defining quadrature-wise density per cell to take advantage of the sub-cell resolution. As shown in numerical experiments, LETO achieves a comparable convergence speed with lower structural compliance.

Optimization and Nonlinear Integrators Another long-standing challenge of topology optimization is to optimize structures undergoing large deformations, requiring a nonlinear elasticity model and the nonlinear equilibrium constraints. This has become increasingly meaningful with the increasing need for material and structural design in soft robotics(Scharff et al., 2019), wearable devices, and even space antennas, *etc*. With large nonlinear deformations, the force equilibrium is more challenging to solve as it often leads to numerical instabilities, and the optimization itself will converge slower.

Numerical integration of partial differential systems can often be reformulated variationally into an optimization problem. These methods can often achieve improved robustness, accuracy, and performance by taking advantage of well-established optimization approaches. Simulation methods are increasingly applying this strategy to simulate both fluid (Batty et al., 2007), and solid (Gast et al., 2015) dynamics, which often enable large time step sizes. The static solve simply corresponds to infinitely large time step size in a dynamic time-stepping point-of-view. Our method also takes advantage of optimization integrators to solve for the static equilibrium to high accuracy robustly.

For nonlinear optimization problems, Newton-type methods are often used because they can deliver quadratic convergence when the intermediate solution becomes close to the local optima. However, when the initial configuration is far from a local optimum, which is often true in static solves, Newton's method may fail to provide a proper search direction as the Hessian can be indefinite (Wang et al., 2019; Liu et al., 2017b,b; Smith et al., 2018). Teran *et al*. (Teran et al., 2005) proposed a positive definite fix to project the Hessian to a symmetric positive definite form to guarantee that a descent direction can be found. This method is referred to as projected Newton (PN) throughout the paper and is applied in the static solve.

7.1.1.3 Artificially Stiff Patterns

Traditional density-based topology optimization methods considering one degree of freedom for density per element may form checkerboard density patterns across elements. Such solutions are indeed optimal mathematically but meaningless in reality. To avoid the checkerboard pattern, filters (Sigmund, 2001; Andreassen et al., 2011) were proposed to smooth the density or gradient field. These filters can be applied independent of the sensitivity analysis or encoded into the objective (Wu et al., 2015), where the latter one defines a consistent optimization problem. In this paper, the SPH kernel-based density transfer between carrier particles and quadrature points serves the same purpose and is explicitly encoded in the objective.

When multiple densities are modeled inside one element to generate higher-resolution details with a relatively low computational cost (Nguyen et al., 2009), sub-cell-level checkerboard issues, also called the QR patterns (Gupta et al., 2018), may appear. Such artifacts happen on a sub-cell level where disconnected interfaces between material components may form inside one element at the quadrature level. However, from the simulation grid view, the disconnected components are connected, producing inaccurate compliance measurement at static equilibrium, which can lead to results with large compliance when tested in practice or simulated with higher resolution. An illustration of the checkerboard issue and the QR pattern issue is shown in Figure 7.1. Existing solutions for avoiding QR patterns include increasing the filter radius of gradient filters (Nguyen et al., 2009), changing penalty power in SIMP formulation (Gupta et al., 2018) and using higher-order elements (Groen et al., 2016). If applied naively, our method may encounter this kind of artifact as well if no treatment is applied since the resolution of MPM quadratures is higher than the background grid. To take advantage of the multi-resolution nature of MPM while keeping the formulation simple, a graph-based narrow-band filter and a connectivity correction algorithm are developed: the set of material points is considered as a graph and only one major connected component is preserved during optimization. A secondary correction is applied after the optimization.

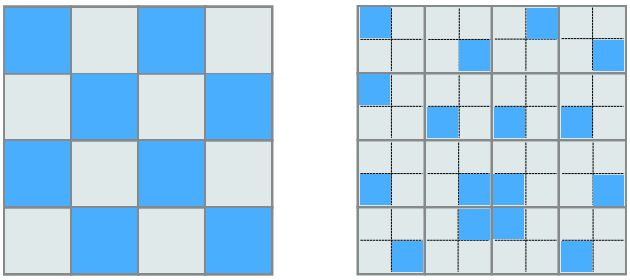


Figure 7.1: An illustration of the checkerboard issue and the QR pattern issue. In both cases, the density fields are continuous from the view of the grids

7.1.1.4 Summary

In summary, a novel hybrid Lagrangian-Eulerian framework LETO is proposed for compliance-based topology optimization with MPM. The hybrid representation of the material density field enables both the flexibility of Lagrangian models and Eulerian methods' computational efficiency. The MPM discretization introduces a multi-density scheme naturally, provides a unified treatment for both linear and nonlinear topology optimization, and supports optimizing fully nonlinear compliance, enabling robust and accurate optimization of structures undergoing large deformations. A graph-based narrow-band filter and a connectivity correction algorithm are applied during and after the optimization to eliminate QR-patterns. The numerical experiments show that the resulting scheme can generate sub-cell structures with mechanical performances that sometimes rival conventional methods at a comparable computational cost.

7.1.2 Problem Statement and Method Overview

7.1.2.1 Problem Statement

The general objective of compliance-based topology optimization is to seek for a material distribution ρ, a scalar field representing the material density at each point on a design domain Ω, to obtain the minimal structural compliance $c(\rho, \mathbf{u})$, or equivalently, the least strain energy $e(\rho, \mathbf{u})$, under force equilibrium between external force load $\mathbf{f}$ and internal

elasticity force $-\frac{\partial e}{\partial \mathbf{u}}$ with displacement $\mathbf{u}$:

$$\min_{\rho} \; c(\rho, \mathbf{u}) = e(\rho, \mathbf{u}) \quad \text{s.t.} \quad \begin{cases} \frac{\partial e}{\partial \mathbf{u}}(\rho, \mathbf{u}) = \mathbf{f} \\ \mathbf{D}\mathbf{u} = \mathbf{0} \\ V(\rho) \leq \hat{V}. \end{cases} \tag{7.1}$$

Here $\mathbf{D}\mathbf{u} = \mathbf{0}$ is the discretized Dirichlet boundary condition where $\mathbf{D}$ selects the zero displacement nodes, $V(\rho) = \int_{\Omega_0} \rho d\mathbf{X}$ is the total volume of the structure, and $\hat{V}$ is an upper bound specified by users to avoid trivial solutions(Sigmund, 2001). Usually, ρ is expected to be close to either 0 or 1 for manufacturing, which potentially makes the problem non-smooth. The density field ρ can be discretized and further parameterized by any set of design variables. For example, the traditional SIMP method assumes that each finite element has a uniform density and directly uses cell densities as design variables. In this paper, a set of movable Lagrangian particles carried with density sources is used as design variables.

The strain energy of the material under a displacement field u is defined as

$$e(\rho, \mathbf{u}) = \int_{\Omega} \Psi(\mathbf{F}) d\mathbf{X}, \tag{7.2}$$

where Ψ is the elastic energy density determined by the underlying constitutive model, and $\mathbf{F}$ is the deformation gradient defined as

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} = \mathbf{I} + \frac{\partial \mathbf{u}}{\partial \mathbf{X}} \tag{7.3}$$

through the world and material space coordinates $\mathbf{x}$ and $\mathbf{X}$ with $\mathbf{u}(\mathbf{X}) = \mathbf{x} - \mathbf{X}$ and $\mathbf{I}$ is the identity matrix. For linear elasticity,

$$\Psi_L(\mathbf{F}) = \mu ||\epsilon(\mathbf{F})||^2 + \frac{\lambda}{2} \text{tr}(\epsilon(\mathbf{F}))^2, \tag{7.4}$$

where $\epsilon(\mathbf{F}) = \frac{1}{2}(\mathbf{F} + \mathbf{F}^T) - \mathbf{I}$ is the small strain, and the Lamé parameters μ and λ linearly relate to the Young's modulus E. However, linear elasticity is only accurate under infinitesimal

deformation since rotation is also penalized, and the nonlinear stress-strain curve is not well captured. A nonlinear constitutive model should be used to model large deformation. In this paper, we adopt the neo-Hookean hyperelasticity model:

$$\Psi_{\mathrm{NH}}(\mathbf{F}) = \frac{\mu}{2}(\mathrm{tr}(\mathbf{F}^T\mathbf{F}) - d) - \mu \log J + \frac{\lambda}{2}(\log J)^2, \tag{7.5}$$

where $J = \det \mathbf{F}$, and $d = 2$ or 3 is the dimension of the problem.

The compliance objective depends on both the density ρ and the displacement u, which is generally nonlinear even for linear elastic materials as well as the static equilibrium constraint. Therefore, the adjoint method (Giles and Pierce, 2000) is often applied to avoid solving the nonlinear Karush-Kuhn-Tucker (KKT) system as in equality constrained optimization (Nocedal and Wright, 1999). It takes $\mathbf{u}$ as a function of $\mathbf{x}$ and cancels out $\frac{\partial \mathbf{u}}{\partial \mathbf{x}}$ by considering the searching process to be conducted only on the force equilibrium constraint manifold. Given an intermediate state ρ, the PDE constraint has to hold by solving the displacement field $\mathbf{u}$ at static equilibrium for each iteration. For linearly elastic materials, finding the static equilibrium results in solving a linear system of equations, which is generally considered the topology optimization's bottleneck. Therefore, obtaining intricate structural features by increasing resolution demands extra computational powers or carefully designed implementations. (Aage et al., 2017; Liu et al., 2018). It becomes even more challenging for nonlinear hyperelastic materials because a nonlinear system of equations needs to be solved at each optimization iteration, leading to numerical instabilities.

7.1.2.2 Lagrangian-Eulerian Multi-Density Topology Optimization

A hybrid Lagrangian-Eulerian approach is proposed to establish a versatile topology optimization framework that can accommodate different elastic models to address the aforementioned challenges. In particular, the elastic potential as the compliance objective is optimized for both linear and highly nonlinear (*e.g.*, neo-Hookean) elastic materials. A set of carrier particles is adopted to represent the material distribution and evolution. Each particle is a moving material sample carrying the information of position $\mathbf{x}^c$, density ρ^c, and supporting radius.

This modifies the general formulation in Equation (7.1) from a pure Eulerian representation, which directly optimizes the density field ρ, to a hybrid Lagrangian-Eulerian form, which jointly optimizes $\mathbf{x}^c$ and ρ^c that define the material distribution $\rho(\mathbf{x}^c, \rho^c)$ over the design domain:

$$\min_{\mathbf{x}^c, \rho^c} \; c(\rho(\mathbf{x}^c, \rho^c), \mathbf{u}) = e(\rho(\mathbf{x}^c, \rho^c), \mathbf{u}) \quad \text{s.t.} \quad \begin{cases} \frac{\partial e}{\partial \mathbf{u}}(\rho(\mathbf{x}^c, \rho^c), \mathbf{u}) = \mathbf{f} \\ \mathbf{D}\mathbf{u} = \mathbf{0} \\ V(\rho(\mathbf{x}^c, \rho^c)) \leq \hat{V}. \end{cases} \tag{7.6}$$

Using MPM as the static equilibrium solver, the design domain is discretized with an Eulerian background grid and a set of uniformly sampled quadrature points in each grid cell where every quadrature has its independent density value. These quadrature points jointly form the scalar field ρ. In this way, a sub-cell resolution is naturally resolved through multiple quadratures per cell. The relation between carrier particles ($\mathbf{x}^c$, ρ^c) and quadrature points (ρ) is further constructed using smoothed-particle hydrodynamics (SPH) kernel and a sharp density mapping function. Adopting the SPH kernel has the equivalent effect as the gradient filter (Sigmund, 2001) that prevents the checkerboard pattern. However, since the SPH kernel is directly incorporated in the objective, it avoids performing any extra smoothing on the gradient, keeping the search direction and the objective consistent.

To efficiently solve the static equilibrium, it is straightforward to achieve a narrow-band sparse simulation (Liu et al., 2018) using MPM by filtering out low-density quadratures. This is essentially equivalent to how zero-mass grid nodes are filtered out in MPM-based dynamic simulations. This mechanism is also adopted to eliminate QR-patterns by maintaining a single main connected component to filter out isolated material blocks. The narrow-band filter threshold's increment replaces the Heaviside projection in traditional density-based topology optimization algorithm for producing binarized designs.

Using moving asymptotes (MMA) (Svanberg, 1987b) as the optimizer, the optimization pipeline can be summarized as the follows; also see an illustration in Figure 7.2.

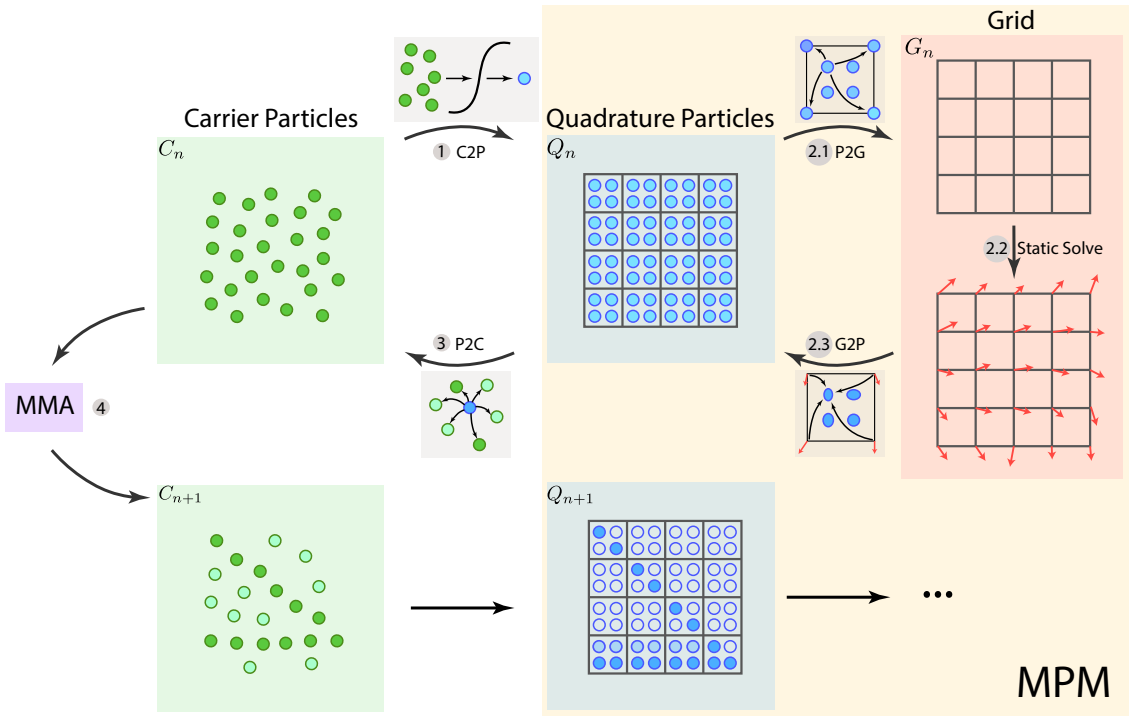


Figure 7.2: **Hybrid Lagrangian-Eulerian method pipeline with an MPM solver.**

0. **Initialize:** Collocate carrier particles on the uniformly sampled MPM quadrature points and initialize carrier particle density ρ^c with the target volume prescription such that the volume constraint is satisfied; see Section 7.1.3.1.

1. **Transfer information from carrier particles to quadrature points (C2P):** Transfer density from carrier particles to quadrature points $(\rho(\mathbf{x}^c, \rho^c))$ with a spherical kernel and a sharp density mapping function; see Section 7.1.3.1.

2. **MPM Static Solve**; see Section 7.1.3.3.

   2.1. **Transfer information from quadrature points to grid (P2G):** Extract the main connected component using the narraw-band filter; see Section 7.1.3.4. Transfer density from quadrature points in the main connected component to grid nodes and construct the MPM system matrix $\frac{\partial^2 e}{\partial \mathbf{u}^2}$ on the grid. Untouched grids are dropped out of degrees of freedom.

   2.2. **Solve force equilibrium:** Solve $\frac{\partial e}{\partial \mathbf{u}} = \mathbf{f}$ for the displacement field $\mathbf{u}$ on the MPM grid subject to Dirichlet boundary conditions $\mathbf{Du} = \mathbf{0}$. Here, only solving

a single linear system is required for material with a linear elastic material (see Section 7.1.3.2). On the other hand, the projected Newton method with line search that guarantees stability and convergence is applied for nonlinearly elastic materials (see Section 7.1.3.3).

2.3. **Update quadrature deformation gradient (G2P):** Update the deformation gradient $\mathbf{F}_q$ of quadrature points with the solved nodal displacement field $\mathbf{u}$.

3. **Compute compliance and the derivatives (P2C):** Evaluate compliance objective $e(\rho, \mathbf{u})$ (Equation (7.25)), compliance derivative $de/d\{\mathbf{x}^c, \rho^c\}$ (Equation (7.19)), volume V (Equation (7.15)), and volume derivative $dV/d\{\mathbf{x}^c, \rho^c\}$ (Equation (7.16)) for optimization search; see Section 7.1.3.2.

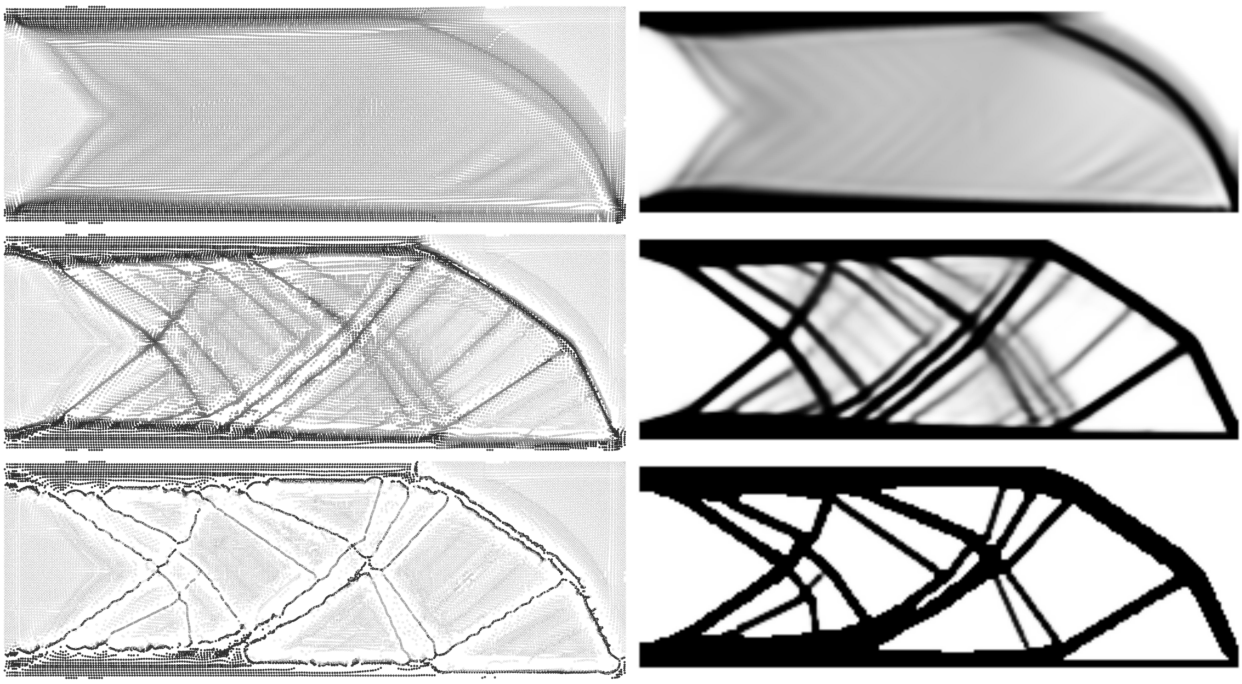
4. **Update carrier particle data:** Update $\mathbf{x}^c, \rho^c$ using MMA and evaluate convergence criteria; see Section 7.1.4.1. If not converged, go to Step 1 and repeat.

5. **Graph-based connectivity correction:** Correct quadrature connectivity according to grid connectivity where the static equilibrium is evaluated; see Section 7.1.3.4.

7.1.3 Hybrid Lagrangian-Eulerian Multi-Density Method

7.1.3.1 Material Distribution Representation

Introducing Lagrangian degrees of freedom by optimizing quadrature positions together with quadrature densities is a straightforward choice. However, arbitrary movements of quadrature may cause large numerical errors, which can even lead to degeneracies like quadrature clustering and isolation. Thus, another set of moving carrier particles is introduced as the design variables to re-parameterize the density field space and, at the same time, to avoid moving quadrature points. Carrier particles are defined in the entire design domain with Lagrangian variables $\xi = (\mathbf{x}^c, \rho^c)$ consists of both position and density. The final material distribution and the volume constraint are still discretized on quadrature points, where their densities are computed according to the surrounding carrier particles. In the proposed method, the carrier particles are crucial in emerging intricate geometry structures (see Figure 7.3).



(a) (b)

Figure 7.3: **Optimization evolution on carrier and quadrature points.** (a) The position and density changes of carrier particles. (b) The density changes on quadrature points.

The density of each quadrature point q is defined as the weighted sum of its neighboring carrier particles $\{\alpha\}$ using an SPH kernel:

$$\tilde{\rho}_q = \sum_\alpha \int_\Omega \rho_\alpha^c W\left(\frac{|\mathbf{x}_\alpha^c - \mathbf{X}|}{h}\right) d\mathbf{X} \approx \sum_\alpha \rho_\alpha^c W\left(\frac{|\mathbf{x}_\alpha^c - \mathbf{x}_q|}{h}\right) V_\alpha, \tag{7.7}$$

where $W(R)$ is a kernel function, h is the kernel size, and V_α is the volume of each quadrature, which equals to $(\frac{\Delta x}{2})^d$ when 2^d quadratures are sampled in each cell. In this paper, the kernel function with cubic spline is applied:

$$W(R) = \sigma \begin{cases} 1 - \frac{3}{2}R^2 + \frac{3}{4}R^3, & 0 < R < 1 \\ \frac{1}{4}(2-R)^3, & 1 < R < 2 \\ 0, & \text{otherwise} \end{cases} \tag{7.8}$$

where σ is a constant of $\frac{10}{7\pi h^2}$ in 2D and $\frac{1}{\pi h^3}$ in 3D.

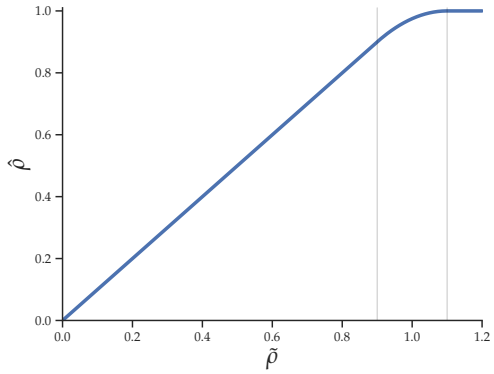


Figure 7.4: Density mapping function.

The derivative of $\tilde{\rho}_q$ w.r.t. the design variables is given by

$$\frac{\partial \tilde{\rho}_q}{\partial \mathbf{x}_\alpha^c} = \rho_\alpha^c \frac{\partial W}{\partial \mathbf{x}_\alpha^c} V_\alpha, \tag{7.9}$$

and

$$\frac{\partial \tilde{\rho}_q}{\partial \rho_\alpha^c} = W\left(\frac{|\mathbf{x}_\alpha^c - \mathbf{x}_q|}{h}\right) V_\alpha. \tag{7.10}$$

To prevent the densities of quadrature particles from exceeding one, a smooth clamping function is further added on top of $\tilde{\rho}_q$ (see Figure 7.4):

$$\hat{\rho}(\tilde{\rho}) = \begin{cases} \tilde{\rho}, & 0 \leq \tilde{\rho} < 1 - \epsilon \\ \frac{(\tilde{\rho}+\epsilon-1)^2}{4\epsilon} + \tilde{\rho}, & 1 - \epsilon \leq \tilde{\rho} < 1 + \epsilon \\ 1, & \tilde{\rho} \geq 1 + \epsilon \end{cases} \tag{7.11}$$

and its derivative is

$$\frac{\partial \hat{\rho}}{\partial \tilde{\rho}} = \begin{cases} 1, & 0 \leq \tilde{\rho} < 1 - \epsilon \\ \frac{(\tilde{\rho}+\epsilon-1)}{2\epsilon} + 1, & 1 - \epsilon \leq \tilde{\rho} < 1 + \epsilon \\ 0. & \tilde{\rho} \geq 1 + \epsilon \end{cases}. \tag{7.12}$$

If $\tilde{\rho}$ is greater than $1 + \epsilon$, the output will be one, and the derivative will be zero. Consequently, through chain-rule, the gradients w.r.t. the design variables vanish, which successfully prevent aggregation of particles. Otherwise, stiffer material than allowed can be

formed by gathering particles, leading to non-physical material.

Finally, the density of each quadrature q is given by,

$$\rho_q = \hat{\rho}(\tilde{\rho}), \tag{7.13}$$

and its derivative is

$$\frac{d\rho_q}{d\xi} = \frac{d\hat{\rho}_q}{d\tilde{\rho}_q}\frac{d\tilde{\rho}_q}{d\xi}. \tag{7.14}$$

The volume of the structure is given by

$$V(\xi) = \sum_q \rho_q, \tag{7.15}$$

where q indices all quadrature points, and its derivative is

$$\frac{dV}{d\xi} = \sum_q \frac{\partial \rho_q}{\partial \tilde{\rho}}\frac{\partial \tilde{\rho}}{\partial \xi}, \tag{7.16}$$

The densities of carrier particles are initialized to a uniform scale such that the density of each quadrature is equal to the prescribed volume fraction (the prescribed volume divided by the domain volume). Since the clamping function is only nonlinear after $1 - \epsilon$, by the partition-of-unity property of SPH kernel, the initial volume is very close to the target volume, where the error comes from the approximation in Equation (7.7). Also note that since the volume constraints are defined on quadrature volumes and the output structure is also represented by quadratures with nonzero density values, there is no need to consider the mass/volume conservation during carrier-quadrature transfers.

7.1.3.2 Design Sensitivity Analysis

To compute the derivatives of the compliance objective e w.r.t. design variables ξ required for the topology optimization, the searching process of finding the adjoint variables is constrained to be solely on the force equilibrium manifold.

$\frac{de}{d\xi}$ can be expressed via applying the chain rule as

$$\frac{de}{d\xi} = \left[\frac{d\rho}{d\xi}\right]^T \left(\frac{\partial e}{\partial \rho} + \left[\frac{d\mathbf{u}}{d\rho}\right]^T \frac{\partial e}{\partial \mathbf{u}}\right). \tag{7.17}$$

Here $\frac{d\mathbf{u}}{d\rho}$ is difficult to compute as $\mathbf{u}$ and ρ are related by the force equilibrium equation; even the evaluation of $\mathbf{u}$ from ρ requires solving a system of equations. However, if the searching process is constrained to be only on the constraint manifold defined by the force equilibrium equation, differentiating $\frac{\partial e}{\partial \mathbf{u}} = \mathbf{f}$ w.r.t. ρ provides

$$\frac{\partial^2 e}{\partial \rho \partial \mathbf{u}} + \left[\frac{d\mathbf{u}}{d\rho}\right]^T \frac{\partial^2 e}{\partial \mathbf{u}^2} = 0, \tag{7.18}$$

then Equation (7.17) can be simplified by combining with Equation (7.18) into

$$\frac{de}{d\xi} = \left[\frac{d\rho}{d\xi}\right]^T \left(\frac{\partial e}{\partial \rho} - \frac{\partial^2 e}{\partial \rho \partial \mathbf{u}} \left[\frac{\partial^2 e}{\partial \mathbf{u}^2}\right]^{-1} \mathbf{f}\right), \tag{7.19}$$

which is the final derivative, where the compliance e can be defined by either linear or nonlinear elasticity.

Simplification Under Linear Elasticity The computation of derivative $\frac{de}{d\rho}$ requires solving a linear system. However, when linear elasticity is applied, it can be simplified to the form widely used in linear topology optimization (Sigmund, 2001).

Specifically, when linear elasticity is utilized, the matrix $\frac{\partial^2 e}{\partial \mathbf{u}^2}$ is constant, so the internal elasticity force is linear w.r.t. $\mathbf{u}$, and the potential e is quadratic w.r.t. $\mathbf{u}$. Namely,

$$\begin{aligned} e &= \frac{1}{2}\mathbf{u}^T \mathbf{K} \mathbf{u}, \quad \mathbf{f} = \frac{\partial e}{\partial \mathbf{u}} = \mathbf{K}\mathbf{u}, \quad \frac{\partial^2 e}{\partial \mathbf{u} \partial \mathbf{u}} = \mathbf{K}, \\ \frac{\partial e}{\partial \rho} &= \frac{1}{2}\mathbf{u}^T \frac{\partial \mathbf{K}}{\partial \rho}\mathbf{u}, \quad \frac{\partial^2 e}{\partial \rho \partial \mathbf{u}} = \mathbf{u}^T \frac{\partial \mathbf{K}}{\partial \rho}, \end{aligned} \tag{7.20}$$

where $\mathbf{K}$ is the stiffness matrix depending on densities ρ. Substituting these equations into

Equation (7.19), it follows that

$$\frac{de}{d\rho} = \frac{1}{2}\mathbf{u}^T\frac{\partial \mathbf{K}}{\partial \rho}\mathbf{u} - \mathbf{u}^T\frac{\partial \mathbf{K}}{\partial \rho}\mathbf{K}^{-1}\mathbf{f} = -\frac{1}{2}\mathbf{u}^T\frac{\partial \mathbf{K}}{\partial \rho}\mathbf{u}, \tag{7.21}$$

which is exactly the same form derived in traditional linear topology optimization via the adjoint method. For arbitrary design variables ξ, applying the chain-rule, the derivative becomes

$$\frac{de}{d\xi} = -\frac{1}{2}\left[\frac{d\rho}{d\xi}\right]^T\left[\mathbf{u}^T\frac{\partial \mathbf{K}}{\partial \rho}\mathbf{u}\right]. \tag{7.22}$$

7.1.3.3 MPM Discretization for Multi-Density Topology Optimization

In MPM, the design domain Ω is discretized with a set of material particles, or quadratures to approximate the integrals in Equation (7.1), which are computed as the weighted sum over all the quadratures:

$$e(\rho, \mathbf{u}) = \int_\Omega \Psi(\mathbf{F})dX \approx \sum_q \Psi(\mathbf{F}_q)V_q. \tag{7.23}$$

$$V(\rho) = \int_\Omega \rho(\mathbf{X})dX \approx \sum_q \rho_q V_q. \tag{7.24}$$

where q indices all quadrature points, and each quadrature q has its own density ρ_q, Young's modulus E_q, deformation gradient F_q, elastic energy density function Ψ_q, and volume V_q. To get a sufficiently aligned density field, 2^d quadrature points in each cell are sampled on a regular lattice structure as illustrated in Figure 7.2. With MPM discretization, multiple densities per cell can be handled in a straightforward manner.

Similarly to SIMP, Young's modulus is scaled by the powered density of the particle: $E_q = \rho_q^p E_0$, where E_0 is the material's Young's modulus, so that the stiffness of the material is continuously varying over the domain according to its distribution. Since Ψ is linear w.r.t. Young's modulus, the compliance can be rewritten as

$$e(\rho, \mathbf{u}) = \sum_q \rho_q^p \Psi_0(\mathbf{F}_q)V_q, \tag{7.25}$$

where Ψ_0 is the energy density function with Young's modulus E_0.

Static Equilibrium While MPM is commonly used for dynamic time-stepping, a static problem formulation based only on MPM spatial discretization similar to the total Lagrangian formulation (de Vaucorbeil et al., 2020) is needed for topology optimization. In such a static setting, there is no inertia effect or time-variant variables, meaning that only the elasticity and external force terms are kept:

$$-\frac{\partial \mathbf{e}}{\partial \mathbf{u}}(\rho, \mathbf{u}) + \mathbf{f} = \mathbf{0}, \tag{7.26}$$

where $\mathbf{f}$ is the external body force load (Neumann boundary condition) defined on the grid nodes. Dirichlet boundary conditions can be defined as

$$\mathbf{D}\mathbf{u} = \mathbf{0}, \tag{7.27}$$

where $\mathbf{D}$ is the selection matrix that extracts the Dirichlet grid nodes. From a variational point of view, this is equivalent to solving the following optimization problem

$$\min_{\mathbf{u}} e(\rho, \mathbf{u}) - \mathbf{u}^T\mathbf{f} \quad \text{s.t.} \quad \mathbf{D}\mathbf{u} = \mathbf{0}. \tag{7.28}$$

In MPM, quadratures are embedded in the background Eulerian grid with a B-spline kernel, meaning that the nodal displacement $\mathbf{u}$ is, in fact, defined on the uniform grid nodes. Since in LETO elastostatic problems are solved without material particle (quadrature points) advection, there will be no cell crossing errors or ringing instabilities for MPM even if linear kernel is used for simplicity:

$$N(x) = \begin{cases} 1 - |x|, & 0 \leq x < 1 \\ 0, & 1 \leq x. \end{cases} \tag{7.29}$$

Here the weight ω_{iq} between grid node location $\mathbf{x}_i$ and quadrature location $\mathbf{x}_q$ is defined by

taking the Cartersian product in all dimensions. For example, in 3D,

$$\omega_{iq}(x_q) = N(\frac{1}{h}(x_{q,1} - x_{i,1}))N(\frac{1}{h}(x_{q,2} - x_{i,2}))N(\frac{1}{h}(x_{q,3} - x_{i,3})); \tag{7.30}$$

and in 2D,

$$\omega_{iq}(x_q) = N(\frac{1}{h}(x_{q,1} - x_{i,1}))N(\frac{1}{h}(x_{q,2} - x_{i,2})). \tag{7.31}$$

The deformation gradient F_q on quadrature q is then related to the surrounding grid nodes i as

$$\mathbf{F}_q = \mathbf{I} + \sum_i \mathbf{u}_i \nabla \omega_{iq}^T, \tag{7.32}$$

which also leads to the elasticity force definition

$$-\frac{\partial \mathbf{e}}{\partial \mathbf{u}} = -\sum_q \rho_q^p V_q^0 \frac{\partial \Psi_0(\mathbf{F}_q)}{\partial \mathbf{F}_q} \nabla \omega_{iq}, \tag{7.33}$$

and the elasticity Hessian (in index notation)

$$\frac{\partial^2 e}{\partial u_{i,\alpha} \partial u_{j,\beta}} = \sum_q \rho_q^p V_q^0 (\nabla \omega_{iq})_\delta \frac{\partial^2 \Psi_0(F_q)}{\partial F_{q,\alpha\delta} \partial F_{q,\beta\omega}} (\nabla \omega_{jq})_\omega, \tag{7.34}$$

where $1 \leq \alpha, \beta, \delta, \omega \leq d$ and d is the spatial dimension. $F_{q,\alpha\beta}$ is (α, β)-th element of the deformation gradient F_q.

Compared with the MPM formulation for dynamic problems, the static formulation can be seen as only solving for a single "time step," and the deformation gradient at previous time step, $\mathbf{F}_q^n$, is just the initial undeformed deformation gradient $\mathbf{F}_q^n = \mathbf{F}_q^0 = \mathbf{I}$.

Static Solve with Projected Newton To solve the equilibrium equation more robustly, the variational form (Equation (7.28)) is minimized with the projected Newton's method (Teran et al., 2005) as outlined in Algorithm 7. Dirichlet boundary conditions are handled by modifying the corresponding entries in the matrix and the right-hand-side to keep the problem unconstrained, which is equivalent to eliminating the Lagrange multipliers in the KKT system with linear equality constraints. The stopping criteria is chosen to be $||\Delta \mathbf{u}||_\infty < \tau = 0.1\Delta \mathbf{x}$.

Algorithm 7 Projected Newton for Solving Static Equilibrium

procedure ProjectedNewton(ρ^j, $\mathbf{f}$, $\mathbf{D}$, τ, $\mathbf{u}$)
  $\Delta\mathbf{u} = \mathbf{0}, \mathbf{u}^0 = \mathbf{0}, i = 0$ // initialize
  do
    $\mathbf{P} \leftarrow \text{projectSPD}(\frac{\partial^2 e}{\partial \mathbf{u}^2})$ // project each local Hessian stencil to SPD (Teran et al., 2005)
    $\Delta\mathbf{u} \leftarrow \mathbf{P}^{-1}(\mathbf{f} - \frac{\partial e}{\partial \mathbf{u}})$
    $\alpha \leftarrow \text{LineSearch}(\mathbf{u}^i, \Delta\mathbf{u})$ // Back-tracking line-search
    $\mathbf{u}^{i+1} \leftarrow \mathbf{u}^i + \alpha\Delta\mathbf{u}$
    $i \leftarrow i + 1$
  while $||\Delta\mathbf{u}||_\infty \geq \tau$
  $\mathbf{u} \leftarrow \mathbf{u}^i$
end procedure

Note that when a linear constitutive model is used, the system is quadratic, so only one iteration is needed.

Inversion-free Line Search Since in each projected Newton iteration, the Hessian has been projected to symmetric positive definite, the search direction $\Delta\mathbf{u}$ is guaranteed to be a descent direction. Therefore, back-tracking line search can ensure $E(\mathbf{u}^{i+1}) < E(\mathbf{u}^i)$ after each $\mathbf{u}$ update, which effectively stabilizes the iterations and improves convergence.

However, for the noninvertible elasticity energy (neo-Hookean), projected Newton does not necessarily ensure no deformation gradient inversion along search direction $\Delta\mathbf{u}$. Hence, following Smith and Schaefer (Smith and Schaefer, 2015), to further prevent inversion of each $\mathbf{F}_q$, a large feasible step size before each line search is solved by finding the minimum of the smallest positive roots of a family of equations

$$\left\{\det(\mathbf{F}_q(\mathbf{u}^i + \beta_q\Delta\mathbf{u})) = \epsilon_q\right\}. \tag{7.35}$$

The line search step size then starts from $\min_q \beta_q$. Here, $\epsilon_q = 0.1\det(\mathbf{F}_q(\mathbf{u}^i))$ is used to avoid numerical rounding errors, which is more robust than solving with $\epsilon_q = 0$.

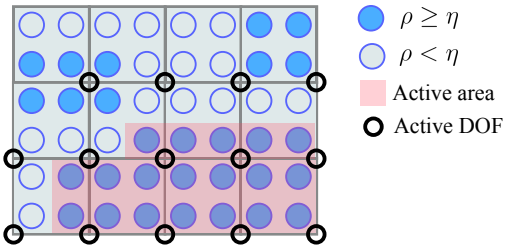


Figure 7.5: An illustration of the narrowband filter mechanism. An active area is first extracted by the graph connectivity. Active DOFs are then indicated by whether they belongs to cells with active quadratures. Only active DOFs are kept for the static solve to reduce the computational cost.

7.1.3.4 Narrow-band Filter and Connectivity Correction

Multi-density topology optimization methods suffer from a common artifact known as QR-patterns. Unlike the traditional checkerboard problem, QR-patterns happen on a sub-cell level, which corresponds to the quadrature here. When QR-patterns appear, there are many sub-cell level isolation of the solid components, which are still viewed as connected from the perspective of the background grid where forces and displacements are discretized. This kind of inconsistency prevents simulation from accurately predicting a structure's compliance, which can lead to results with large compliance when tested in practice or simulated on a grid with higher resolution. Based on this observation, this paper proposes a narrow-band filter to keep only one major connected component during the optimization and then apply a connectivity correction step to correct the final topology further.

Given a threshold η, a graph is built at each optimization iteration before the static solve, with its vertices being those quadrature points having a density greater than η. Quadrature pairs are identified to be adjacent only if their Manhattan distance is 1 (adjacent along one of the coordinate axes). By performing a breadth-first search on this graph, the connected component Θ with the most quadrature points are then extracted. Only the grid nodes within the kernel range of active quadrature points are kept as DOFs in the MPM static solve step. An illustration is shown in Figure 7.5.

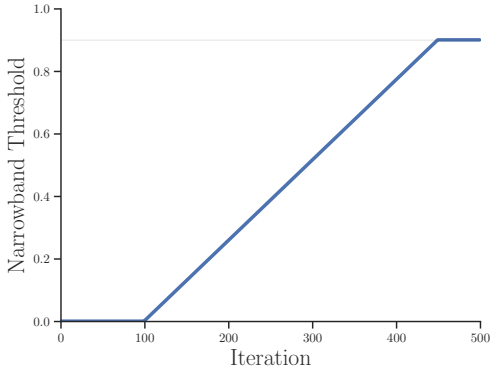


Figure 7.6: Evolution of narrowband threshold.

The narrowband filter θ employed is introduced as:

$$\theta(\rho_q) = \begin{cases} 0, & q \in \Theta \\ 1, & q \notin \Theta, \end{cases} \tag{7.36}$$

with which the topology optimization becomes:

$$\min_{\xi,\mathbf{u}} \; e(\rho, \mathbf{u}) = \sum_q \theta(\rho_q)\rho_q^p \Psi_0(\mathbf{F}_q)V_q, \quad \text{s.t.} \quad \begin{cases} \frac{\partial e}{\partial \mathbf{u}}(\rho, \mathbf{u}) = \mathbf{f} \\ \mathbf{D}\mathbf{u} = \mathbf{0} \\ V(\rho) \leq \hat{V}. \end{cases} \tag{7.37}$$

To handle the non-smoothness of θ,an alternating optimization style strategy is performed that computes θ between each MMA optimization iteration where θ is then treated as constant. Theoretically, this strategy can lead to different local optimum compared to a fully coupled search, but it has been shown to be effective in our experiments. During the optimization, the narrow-band filter can also accelerate the pruning of redundant branches, for example, a fiber with only one end connected to the major component, which can increase the compliance.

Given that the narrow-band filter can remove all quadrature points with a density below η, a new binary design enforcement mechanism is proposed. A low threshold of the narrow-band filter is chosen initially for a coarse but more global result and is further increased per iteration towards a value very close to 1 (Figure 7.6). This mechanism provides better optimization

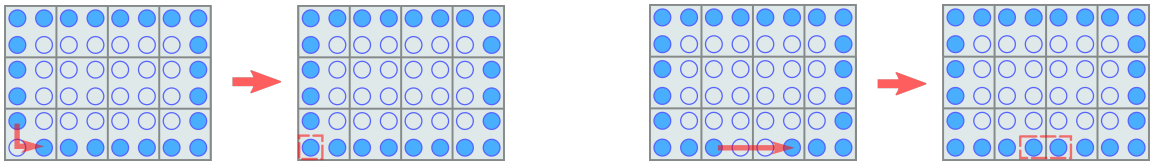


Figure 7.7: Illustrated examples of emerging QR-patterns and connectivity correction procedures.

stability by enforcing binary design without any nonlinear projections, e.g., a Heaviside function.

However, QR-patterns may still exist when the two parts of the major component's boundaries are spatially close. Figure 7.7 lists some illustrated examples. Here the filled and unfilled circles are used to discriminate solid and void materials. It can be seen that despite a notable disconnection of quadrature points, when evaluating the static force equilibrium, these are still considered connected since they can belong to the same cell (the left example) or adjacent cells (the right example). However, when evaluating the resulting design, a double refined grid where each quadrature corresponds to a different cell, the corresponding cells of these separated quadratures will only share one node or are completely disconnected. To detect and remove these QR-patterns, a connectivity correction step is further introduced: for any pair of quadratures belonging to the same cell or two different cells sharing at least one node, if the difference between their distance on the graph (the length of the shortest path) and their spatial L_1 distance is larger than a threshold, a shortest Manhattan path will be created between them with the densities along the path all set to 1. Note that the occurrence of such corrections is rare.

7.1.4 Numerical Examples

7.1.4.1 Optimizing Structures with MMA

The method of moving asymptotes (MMA) (Svanberg, 1987b) is used to solve the optimization problem, which jointly optimizes the positions and densities of carrier particles. This optimizer is designed for general structural optimization problems with inequality constraints and box constraints. The algorithm approximates the original problem with a series of separable convex

Table 7.1: Compliance value and volume percentage for linear elastic experiments. The compliance value of results from the proposed method under optimization grid resolution, under double-refined grid resolution and SIMP method under double-refined grid resolution is shown in this table.

Experiment	LETO				SIMP (r=1.5)		
	Compliance	**Compliance (refined grid)**	Volume	Ave. Cost (per iter.)	**Compliance (refined grid)**	Volume	Ave. Cost (per iter.)
Concentrated-load beam	1.218×10^{-3}	1.243×10^{-3}	29.9%	0.405s	1.264×10^{-3}	30.1%	0.387s
Michell truss	4.736×10^{-2}	5.054×10^{-2}	20.1%	0.544s	5.055×10^{-2}	20.2%	0.355s
Distributed-load beam	7.845×10^{-3}	8.386×10^{-3}	39.7%	0.867s	1.658×10^{-2}	40.0%	0.408s
3D beam	3.323×10^{-4}	3.334×10^{-4}	19.6%	3.753s	3.527×10^{-4}	20.0%	6.110s
3D bridge	1.653×10^{-2}	1.643×10^{-2}	19.4%	6.893s	1.892×10^{-2}	20.0%	16.816s

optimizations. At each iteration, it sets up two asymptotes for each variable to constrain the searching interval. These asymptotes will be updated according to each sub-optimum. In our implementation, we adopt an open-source C++ version of MMA.[2]

To obtain high-quality results from MMA, careful parameter tuning is necessary, and different examples might have different sets of optimal MMA parameters. There are three parameters to tune: $asyinit, asyincr$, and $asydecr$. In this paper, these parameters are set to be $0.02, 1.05$, and 0.65, respectively, as in the original MMC method. The parameters are used throughout all the numerical examples. To further stabilize and accelerate the optimization in a consistent way, the following regularizations are additionally applied within MMA :

1. The step length of each variable is controlled by modifying its box constraint at each iteration. The change of carrier density is controlled to be below 0.5, and the change of carrier position at each dimension is controlled to be below 2 times the background grid spacing. This step size control can stabilize optimization.

2. Following the MMC method, the gradient of objective and the volume constraint are scaled such that their L^∞-norms are both 1. In addition, the objective and the volume constraint are scaled accordingly to make sure the scaled gradients are consistent. The scaling accelerates the optimization significantly.

[2] https://github.com/jdumas/mma

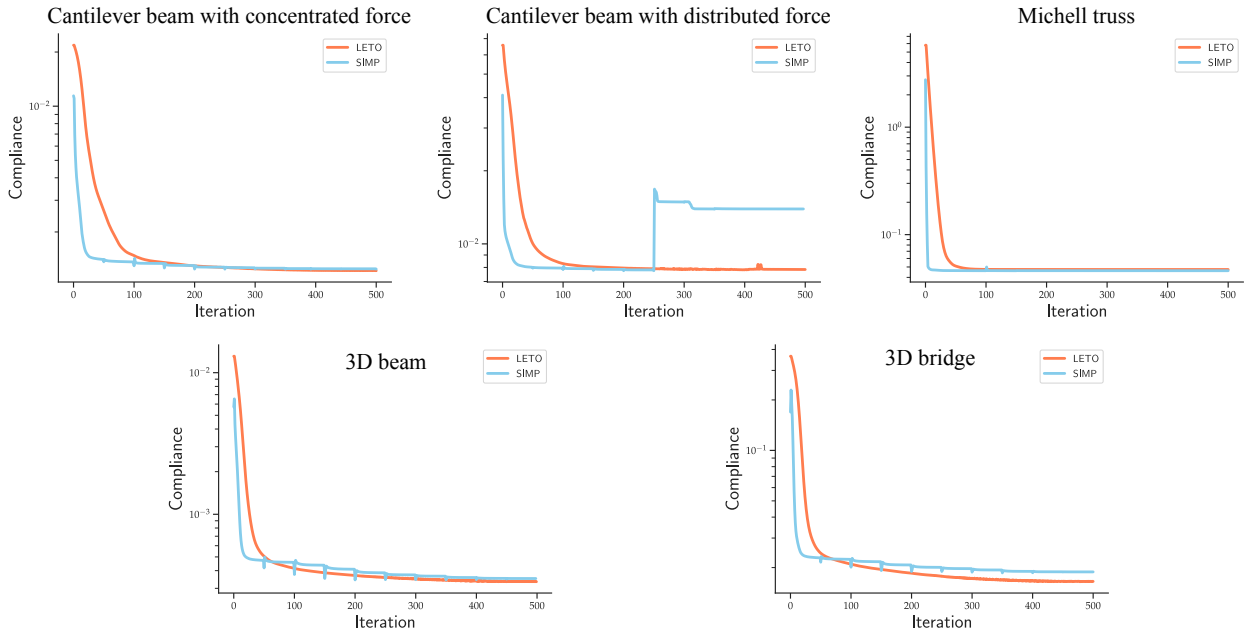


Figure 7.8: Convergence plots for linear elasticity experiments. It can be seen here in the 5 experiments conducted, LETO achieves comparable structural compliance value and convergence speed. Notably, under distributed force condition, LETO delivers better stability.

7.1.4.2 Linear Topology Optimization

In this section, the proposed method is compared with SIMP with Heaviside projection(Guest et al., 2004) on linear elasticity examples. The filter radius of SIMP is 1.5 for all examples. Under the same simulation resolution, LETO obtains more detailed geometry structures and delivers comparable or even lower compliance at a similar convergence speed (see Figure 7.8). For fair comparisons, LETO and SIMP's final results are compared on a double refined grid using FEM, where individual quadrature of LETO corresponds to a cell on the refined grid. Each cell of SIMP is mapped to a 2×2 (in 2D) or $2 \times 2 \times 2$ (in 3D) cell block with the same density. As varying volumes can easily lead to compliance changes, to further ensure fairness, the final volume constraint of LETO is controlled to be slightly lower than the corresponding volume in the SIMP method. For each of the following experiments, SIMP is first tested with the target volume. The volume constraint of LETO is then set to be less than the volume of the binarized SIMP's result. Since LETO treats volume constraints as inequalities through MMA, it usually ends up with an even slightly smaller volume. The binarization threshold of SIMP is set to be 0.5. Since LETO can guarantee there exists no density values

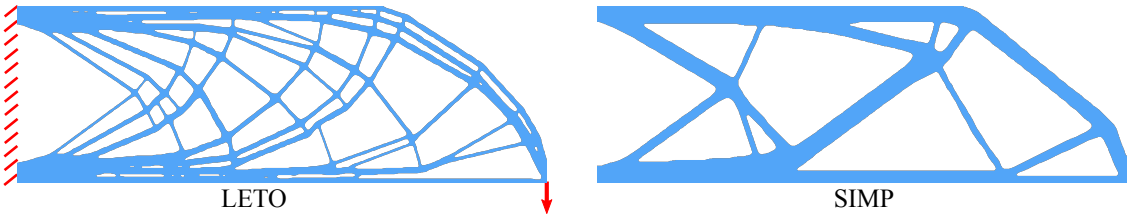


Figure 7.9: **2D beam with a concentrated load.** The comparison between LETO and SIMP on this standard 2D beam example where left most grid nodes are fixed, and a concentrated load is applied at the bottom-right corner. The compliance value evaluated at double-refined grid resolution for LETO and SIMP are 1.243×10^{-3} and 1.264×10^{-3} respectively.

between 0 and 0.9, the binarization threshold of LETO is chosen to be 0.9. To demonstrate that QR-patterns do not appear in the proposed method's final results, compliance values are evaluated at both the resolution used in optimization and with double-resolution. A collection of resulting compliance value and volume percentage of 5 experiments is shown in Table 7.1. The average computational costs per iteration are also shown. Since MMA used in LETO is usually slower than OC used in SIMP, it is expected that LETO is slower than SIMP. However, the narrowband filter can prune inactive DOFs, which can reduce the computational cost a lot after the structure becomes sufficiently binarized (especially in 3D).

2D Beam with a Concentrated Load In this example, a standard 2D beam benchmark problem is tested; see Figure 7.9. The rectangular design domain is $1m$ in width and $3m$ in length, discretized by a 300×100 grid with a spacing of $0.01m$. A 0.1N downward force is concentrated at the bottom-right grid node (denoted by the red arrow) and the leftmost grid nodes are fixed. The target volume constraint is 30%. The optimal material distribution obtained from LETO has rich branching fibers. Even the thick fibers on the boundary of SIMP's result splits into several thin fibers in LETO . The compliance of LETO's result evaluated at optimization resolution and double resolution, and SIMP's result evaluated at double resolution are 1.218×10^{-3}, 1.243×10^{-3}, and 1.264×10^{-3} respectively. The final volume reached by LETO and SIMP are 29.9% and 30.1%.

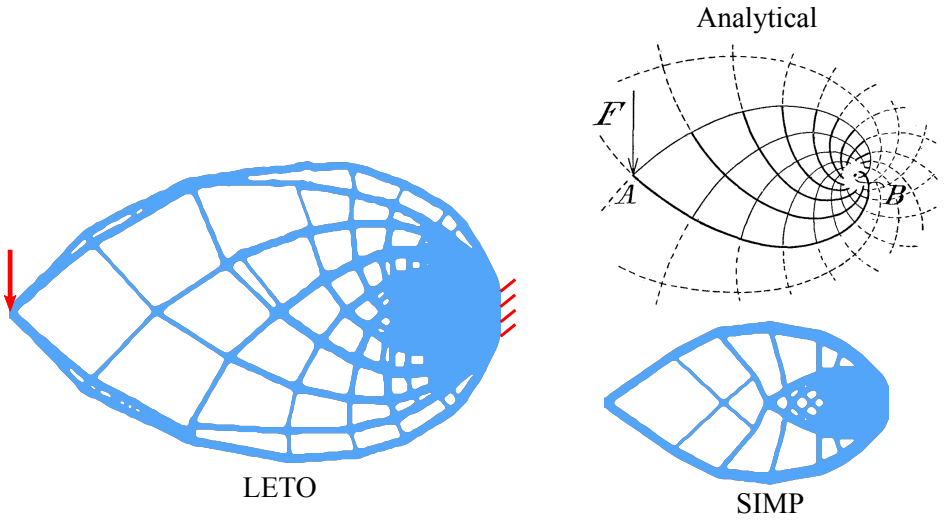


Figure 7.10: **Michell truss.** Here the resulting structure of LETO and SIMP for Michell Truss is evaluated. The central region of the right-most grid is fixed, and a concentrated force is applied in the middle of the leftmost grid. The compliance value evaluated at double-refined grid resolution for LETO and SIMP is 5.054×10^{-2} and 5.055×10^{-2} respectively. The analytical solution (Michell, 1904) is shown for reference, albeit the compliance value of both methods are not significantly different, the fiber directions from the proposed method align with those appeared in the analytical solution better.

Michell Truss The second example under consideration is Michell truss; see Figure 7.10. A $2m \times 1.6m$ rectangle is used as the design domain. The grid resolution is 200×160 with a spacing of $0.01m$. A concentrated downward force of $1N$ is applied to the middle node of the leftmost column of the grid. The middle region of the right-most grid is set to be fixed. The target volume is 20%. The compliance of SIMP evaluated at double resolution, LETO evaluated at optimization resolution, and double resolution are 5.055×10^{-2}, 4.736×10^{-2}, 5.054×10^{-2} respectively. Although LETO's result has almost the same compliance as SIMP, it resembles more to the analytical solution (Michell, 1904) as can be seen from the resulting fiber direction. The final volume reached by SIMP and LETO is 20.2% and 20.1%.

2D Beam with a Distributed Load In this example, the proposed version of LETO and LETO without narrow-band filter (simulate with weak material filling across the whole domain) are also compared to demonstrate the capability of the proposed narrow-band mechanism to remove QR patterns; see Figure 7.11. To enforce binary design when threshold-increasing narrow-band filter is absent, a family of density mapping functions are used like

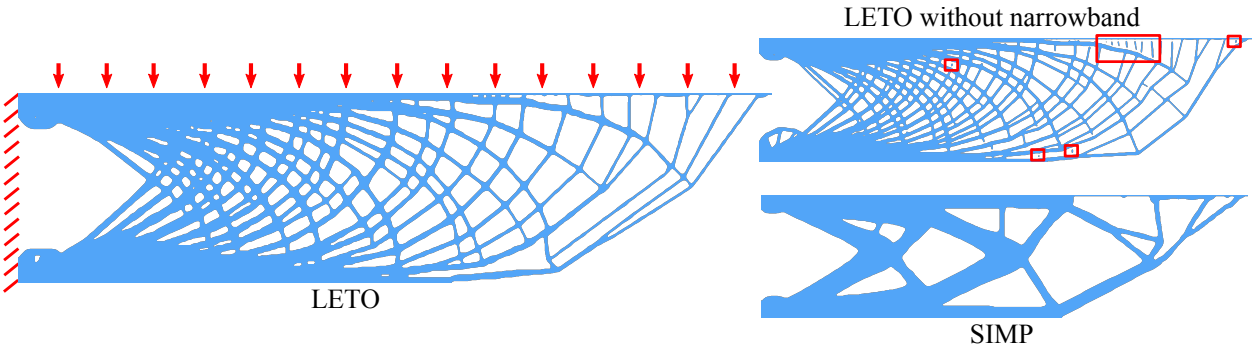


Figure 7.11: **2D beam with a distributed load.** A distributed force load is applied to this beam example at the top plane where the leftmost nodes are fixed. The compliance value evaluated at double-refined grid resolution for LETO and SIMP are 8.386×10^{-3} and 1.658×10^{-2} respectively. LETO without a narrow-band filter is visualized in the top-right corner to demonstrate that the isolated blobs can be removed with the proposed filtering technique.

Heaviside projection in SIMP, except that the functions are smoothly clampped at 0 and 1 to have zero gradient:

$$\hat{\rho}_k(\tilde{\rho}) = \begin{cases} \frac{1}{2}(2\tilde{\rho})^k, & 0 \leq \tilde{\rho} < \frac{1}{2} \\ 1 - \frac{1}{2}(2 - 2\tilde{\rho})^k, & \frac{1}{2} < \tilde{\rho} < 1 \\ 1, & \tilde{\rho} \geq 1, \end{cases} \tag{7.38}$$

where k is gradually increased from 1.01 to 10 during the optimization.

The design domain is a $4m \times 1m$ rectangle. The left boundary is fixed and a total force of $4N$ is evenly distributed on the top boundary. The grid resolution is 400×100 with a spacing $0.01m$. The target volume is 40%. As highlighted by red boxes, a lot of isolated material blobs form when LETO is used without the narrow-band filter. Such visually isolated material blobs actually belong to one continuum at the simulation resolution but are disconnected at quadrature level (or higher resolution). The large compliance difference between evaluations at low resolution ($c = 6.016 \times 10^{-2}$) and high resolution ($c = 7.804 \times 10^{-3}$) also indicates the inconsistency, which differ by an order of magnitude. The proposed version of LETO can remove those isolated blobs automatically during the optimization, resulting in consistent compliance values when evaluated at optimization resolution $c = 7.845 \times 10^{-3}$ and double resolution ($c = 8.386 \times 10^{-3}$). The compliance of SIMP at double resolution is 1.658×10^{-2}.

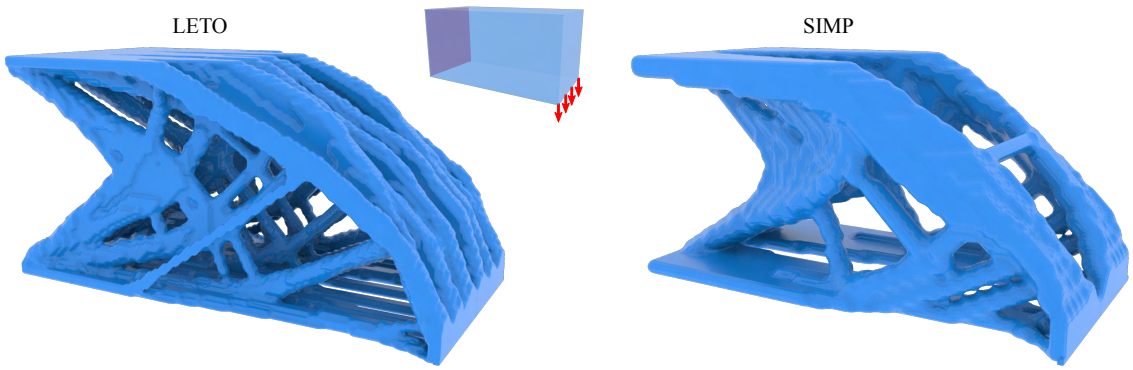


Figure 7.12: **3D beam.** The results of SIMP and the proposed method LETO for this 3D beam example is evaluated. One ending plane of this cubic domain is fixed and forces are applied at the opposite plane's bottom edge. The compliance value evaluated at double-resolution grid for LETO and SIMP is 3.334×10^{-4}, 3.527×10^{-4} respectively. Richer thin supporting fibers can be seen in the result from the LETO .

The final volume reached by SIMP and LETO is 40.0% and 39.7%. Results generated by LETO contains richer intricate fibers than the one by SIMP.

3D Beam Here LETO is further evaluated on a 3D beam problem; see Figure 7.12. In this example, the design domain is a cuboid of $1.6m \times 0.8m \times 0.8m$. One end of this cubic domain is fixed and a total force of $0.825N$ evenly distributes on the bottom edge of the opposite end. A symmetric boundary condition is utilized to reduce the simulation domain to half to save computational cost. The actual simulation grid resolution is $64 \times 32 \times 16$ with a spacing of $0.025m$. The target volume prescription is 20%. With LETO , the result forms a set of thin supporting structures. The compliance for SIMP evaluated at double resolution, LETO evaluated at optimization resolution, and double resolution are 3.527×10^{-4}, 3.323×10^{-4}, 3.334×10^{-4} respectively. The final volume reached by SIMP and LETO is 20.0% and 19.6%.

3D Bridge In the second 3D example, LETO is tested on a 3D bridge problem; see Figure 7.13. The design domain is a cuboid of a length of $4m$, a width of $1m$, and a height of $1m$. The two ending planes along the longest axis are fixed, and a plane force of a total $40.74N$ is added on the bottom (denoted by red arrows). Two symmetric boundary conditions are utilized to reduce the simulation domain to a quarter. The grid resolution in optimization is $160 \times 40 \times 40$ with a spacing of $0.025m$. The target volume is 20%. The appearance of

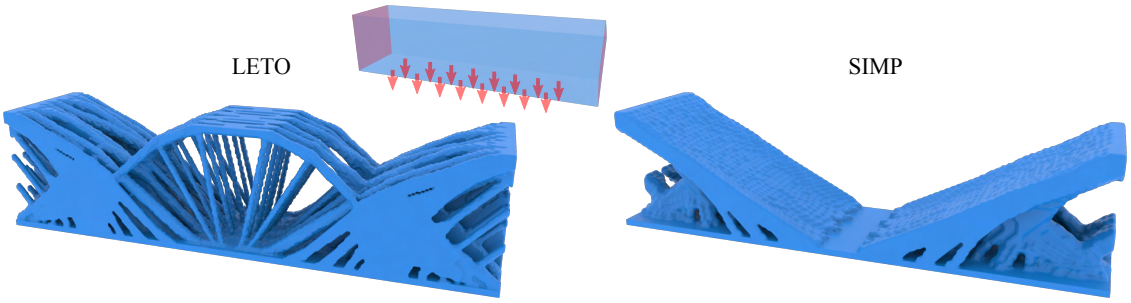


Figure 7.13: **3D bridge.** A 3D brigade design is considered in this example, where the two ending planes of the cubic design domain is fixed, and a plane force is applied at the bottom. The compliance value evaluated at double-refined grid resolution for LETO and SIMP is 1.643×10^{-2}, 1.892×10^{-2} respectively. LETO produces intricate supporting truss structures unseen in the results from the SIMP method.

LETO and SIMP appears to be significantly different in this example, where LETO generates truss structures with rich fibers in the middle. The compliance value for SIMP evaluated at double resolution, LETO evaluated at optimization resolution, and double resolution are 1.892×10^{-2}, 1.653×10^{-2}, 1.643×10^{-2} respectively. The final volume reached by SIMP and LETO at double resolution grid are 20.0% and 19.4%.

Ablation Studies In this section, an ablation study on LETO with different quadrature point distributions, together with a comparison between LETO and SIMP with 1.2 filter radius are demonstrated.

First, two different variations of LETO are compared with the proposed one:

- **LETO with Gaussian quadratures.** The quadratures within each cell are moved to Gaussian points. This means the sample points of the density field are not uniformly distributed as well.
- **Single-density LETO.** The density field within each cell is a constant. The enforcement is achieved by setting each cell's density as the average density of the uniformly-distributed quadratures within that cell. When computing the density field, Gaussian quadratures are utilized to increase the numerical accuracy.

The experiment setup up is the same as Section 7.1.4.2. The optimization results are

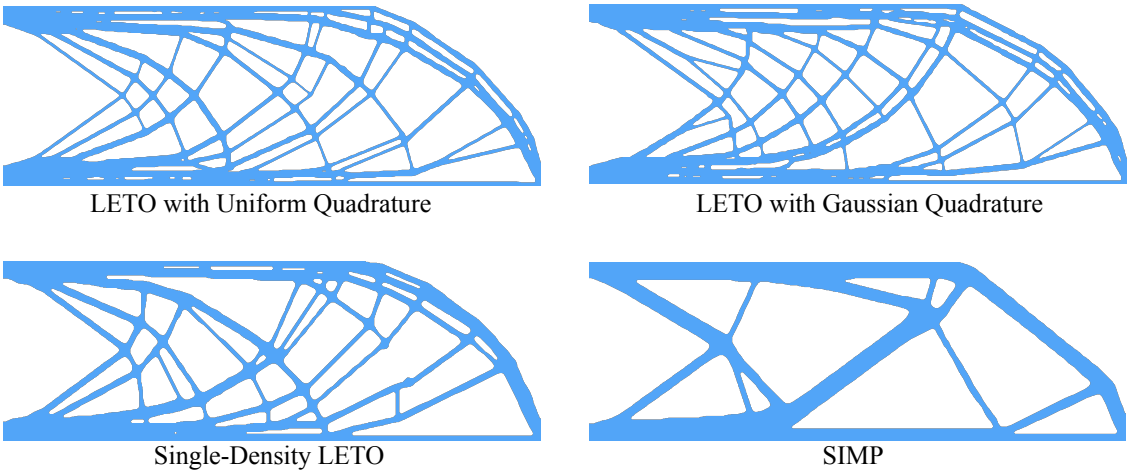


Figure 7.14: Ablation study on variations of LETO.

Method	Quadrature System	Constant Density in Cell	Compliance (refined grid)	Volume
LETO	Uniform	False	1.241×10^{-3}	30.1%
LETO	Gaussian	False	1.257×10^{-3}	30.2%
LETO	Gaussian	True	1.408×10^{-3}	30.5%
SIMP	Gaussian	True	1.264×10^{-3}	30.1%

Table 7.2: Quantative comparison between variations of the proposed LETO.

shown in Figure 7.14, where the top-left one is from the proposed LETO. The top-right one and the bottom-left one are from the two variations of LETO. The bottom right one is from SIMP. The quantitative comparison is included in Table 7.2.

Among these versions of LETO, the proposed one (LETO with uniformly-distributed quadratures) performs the best. Compared to uniformly-distributed quadratures, Gaussian quadratures may cause bias on the sampling of the density field during the C2P transfer. The quantatitive comparison also shows the advantage of uniformly-distributed quadratures. The uni-density version of LETO performs worst on this example. However, a notable distinction between the result from uni-density LETO and the result from SIMP is that the result from uni-density LETO has far more fine structures than SIMP. Likewise, the result from the proposed LETO has more fine structures than uni-density LETO. These comparisons show that the multi-density quadrature system and the particle-based material representation jointly contributes to the generated fine structures.

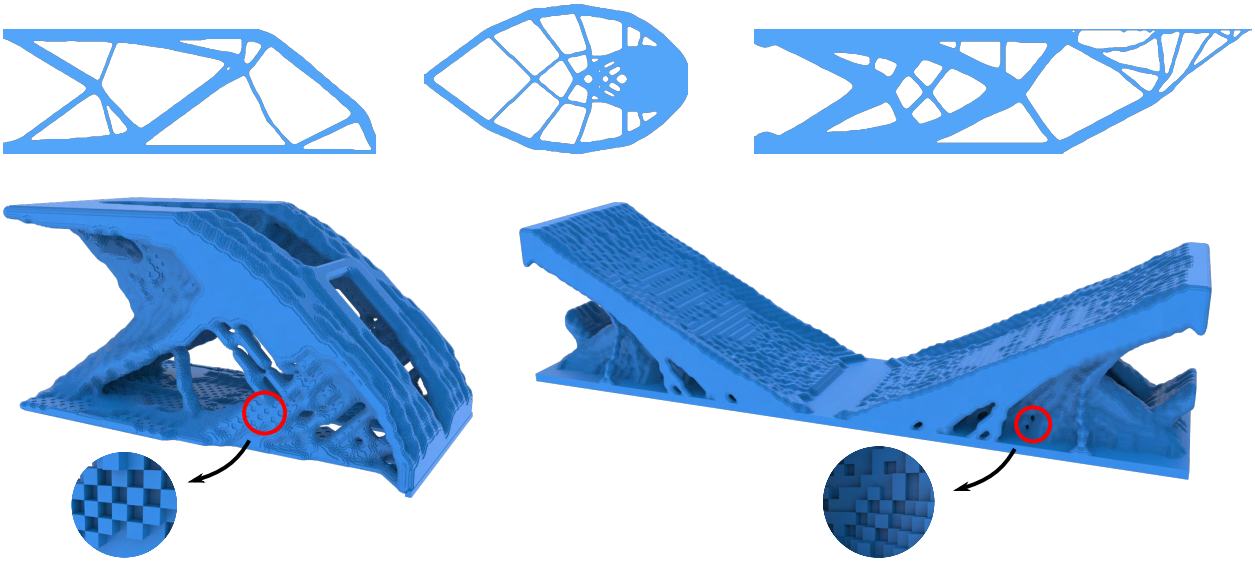


Figure 7.15: Linear topology optimization results of SIMP with filter radius 1.2.

Then LETO is also compared with SIMP with filter radius 1.2. In this case, the filter only touches the adjacent cells on three axes. The results of SIMP on the five linear topology optimization examples are shown in Figure 7.15. The quantitative comparisons are shown in Table 7.3. LETO still performs better in these examples and generate finer structures. In addition, 3D results from SIMP have checkerboard artifacts because of the small filter radius.

Experiment	LETO		SIMP (filter radius = 1.2)	
	Compliance (refined grid)	Volume	Compliance (refined grid)	Volume
Concentrated-load beam	1.243×10^{-3}	29.9%	1.316×10^{-3}	30.0%
Michell truss	5.054×10^{-2}	20.1%	5.055×10^{-2}	20.1%
Distributed-load beam	8.386×10^{-3}	39.7%	1.458×10^{-2}	40.0%
3D beam	3.334×10^{-4}	19.6%	3.666×10^{-4}	20.0%
3D bridge	1.643×10^{-2}	19.4%	1.938×10^{-2}	20.0%

Table 7.3: Quantitative comparison between LETO and SIMP with filter radius 1.2.

7.1.4.3 Nonlinear Topology Optimization

In this section, the robustness of LETO's nonlinear static equilibrium solver is illustrated by varying the force magnitude in large ranges. Two different objectives are also compared: the elastic potential energy and mean compliance $u^T f^{ext}$, where f^{ext} is the nodal external

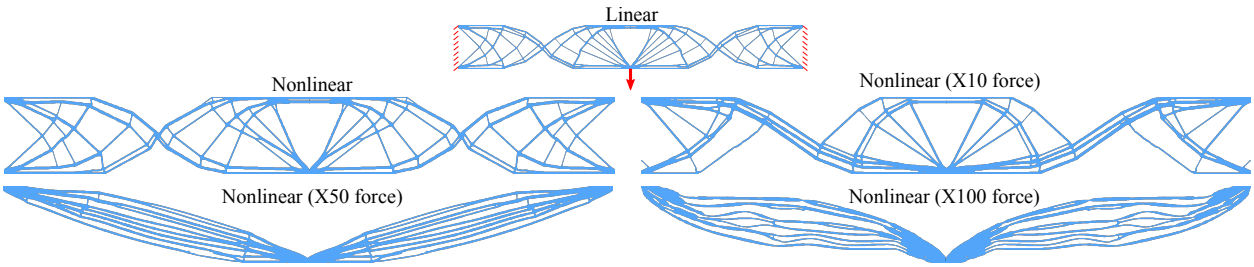


Figure 7.16: **2D long beam.** The results of the proposed method under different force magnitudes are shown in this figure. The resulting material distribution differs sharply and a clear buckling behavior manifested. It can be seen that the proposed method is robust under large deformations.

force field and u is the nodal displacement field. Many topology optimization methods for nonlinear elasticity only consider cases with small strains, thus utilize the mean compliance as the objective function, essentially a linearization of the elastic potential (Buhl et al., 2000; Bruns and Tortorelli, 1998; Gea and Luo, 2001). The two objectives are equivalent in linear elasticity up to a factor. The following examples show that they differ significantly under large force magnitude. Minimizing the mean compliance is equivalent to minimizing the displacements at force loading points. However, that does not necessarily minimize the elastic energy stored in the material. Buckled structures will appear to reduce potential energy when force magnitude is large. The examples shown are all optimized with nonlinear LETO . The compliance and mean compliance value reported are all evaluated on a double refined grid as in the linear topology optimization examples.

2D Long Beam In this example, a concentrated force is applied at the bottom center of a long beam; see Figure 7.16. The $4m \times 1m$ design domain is discretized with a grid resolution 800×100. The target volume is 20%. Force magnitude at $1N, 10N, 50N, 100N$ are tested. As shown in Figure 7.16, when the force magnitude is small, the result is close to that of linear topology optimization. As the force becomes larger, more and more fibers form between the force port and the two top corners. And finally, a lot of buckled fibers appear.

Under the maximal tested force magnitude, results of compliance (elastic potential energy) minimization and mean compliance minimization are compared; see Figure 7.17. The undeformed and deformed states are differentiated by translucent and solid coloring.

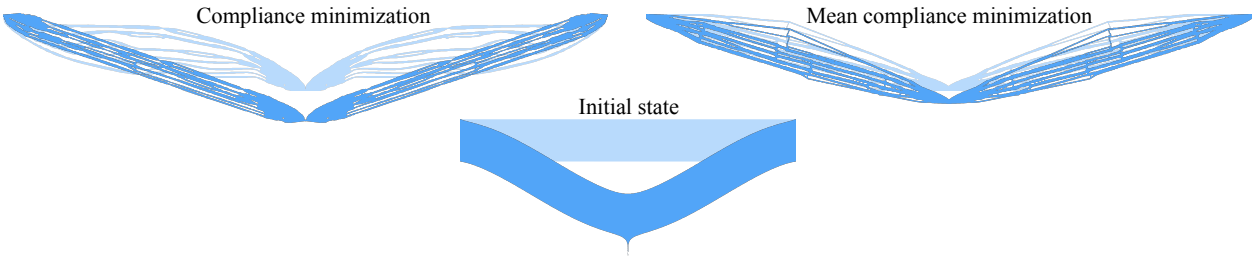


Figure 7.17: **2D long beam structure with different objectives.** The resulting material distribution of minimizing compliance and mean compliance under X100 force is shown in this figure.

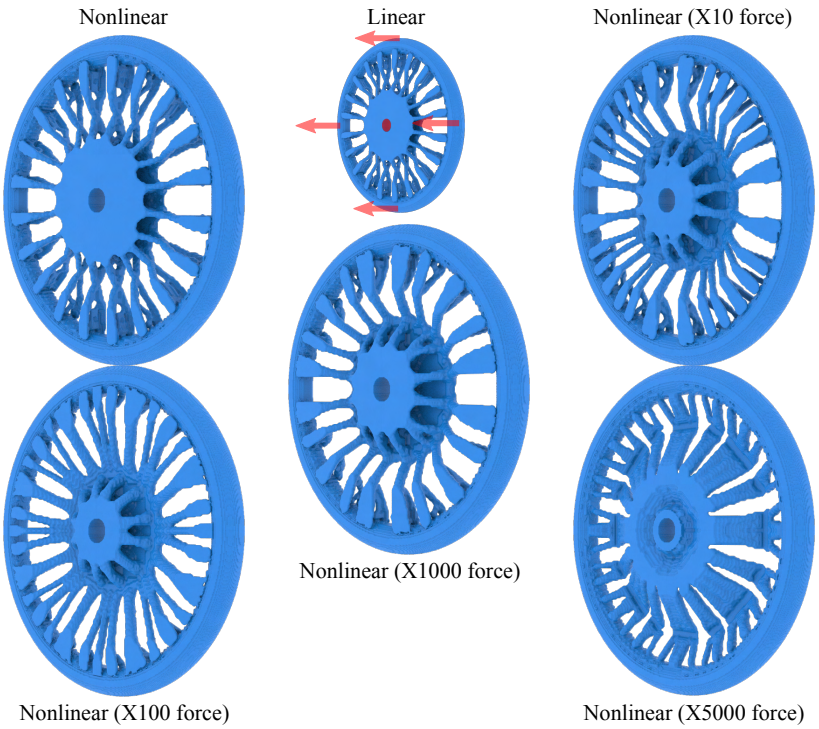


Figure 7.18: **3D wheel.** In this example, a large range of force magnitudes is tested to further examine the robustness of the proposed method in 3D. Out-of-plane forces are applied to the wheel and the central region is fixed.

The final compliance of the result by minimizing elastic potential is 3.473 and the mean compliance of the result by minimizing mean compliance is 3.921. Although the displacement of force port is much smaller when minimizing mean compliance, the elastic energy stored in the structure is much larger: the compliance of the result by minimizing mean compliance is 8.535, which is significantly larger than the compliance value resulting from minimizing the elastic potential.

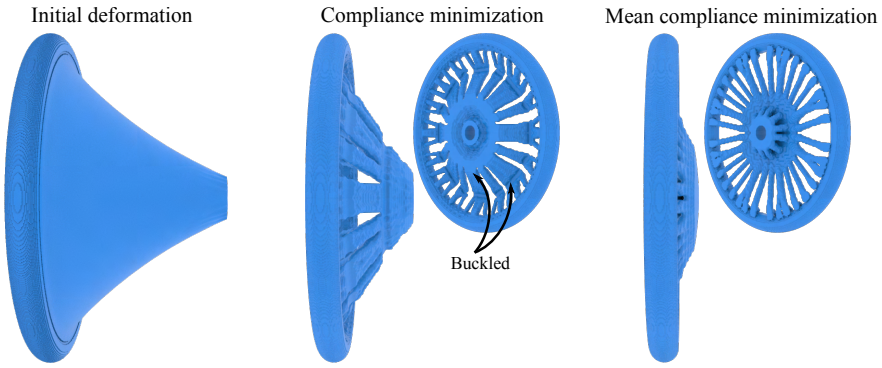


Figure 7.19: **3D wheel structure with different objectives.** The difference in considering compliance and mean compliance as the objective function is further compared in this experiment, where a force is enforced to create significantly large initial deformation. Minimizing compliance objective delivers strong buckling behavior in the resulting structure.

3D Wheel In this example, LETO is tested on a 3D wheel design problem under normal forces; see Figure 7.18. The design domain is a flat cylinder bounded by a torus. The outer radius and inner radius of the torus are $1.2m$ and $1m$, respectively. The grid resolution is $96 \times 96 \times 8$ with spacing $0.025m$. A small cylinder through the center of the wheel is fixed, and the outer torus is set to maintain solid. Forces are perpendicular to the wheel plane and are evenly exerted on a thin layer of the wheel's outer-most boundary. The magnitude of total force at $1.09 \times 10^{-2}N, 1.09 \times 10^{-1}N, 1.09N, 1.09 \times 10^{1}N, 5.44 \times 10^{1}N$ are tested. The target volume is 20%. When the force magnitude is small, the result of nonlinear elasticity is almost identical to linear elasticity, which is symmetric w.r.t the wheel plane. As the force magnitude becomes larger, the symmetry disappears, and the spokes become denser.

Same as the 2D example, when the force is very large, buckled fibers appear. As shown in Figure 7.19, the buckled spokes open up in the deformed state. Result optimized by minimizing mean compliance is shown as well to compare. Although the overall deformation of it appears to be more moderate than the compliance-minimized result, it actually stores significantly more energy. The mean compliance of the mean-compliance-minimized result is 2.946. However, the energy it contains is 6.960, while the compliance of the compliance-minimized result is only 2.063.

7.1.5 Conclusion and Future work

A new hybrid Lagrangian-Eulerian topology optimization method is proposed. MPM discretization is used to enable sub-cell resolution on the fly. The method produces intricate results with comparable and sometimes lower compliance at similar simulation costs than Eulerian methods. With a unified treatment, the proposed method optimizes the elastic potential as the compliance objective for both linear and highly nonlinear (*e.g.* neo-Hookean) hyperelastic materials. Notably, the method robustly captures large deformation and buckling behaviors in nonlinear cases.

The finite strain formulation in MPM allows the construction of a unified framework for general hyperelastic materials. With the ability to resolve sub-cell features, LETO can be further extended in future work to optimize anisotropic, heterogeneous, and multi-scale materials. It would also be interesting to apply this framework to optimize different objectives, *e.g.*, compliant mechanisms and task-oriented objectives for designing soft robots.

In the 2D long beam example for nonlinear elasticity, slight overlaps between some fibers happen under the maximal tested load when the equilibrium is solved with the narrow-band filter turned on. However, when the equilibrium is solved with weak material like SIMP, the non-invertible neo-Hookean constitutive model will push fibers apart. This brings the inspiration that non-invertible weak material could be a contact handling model. On the other hand, it is tricky to tune weak material's Young's modulus — too large Young's modulus can provide non-realistic supporting force. On the other hand, too small Young's modulus may cause numerical issues, not to mention that the weak material wastes a lot of computational power. More robust handling of contact should be developed to enable contact-aware topology optimization.

Last but not least, the method relies on MMA. As mentioned, MMA requires careful parameter tuning to perform well. Even with the general and consistent regularizations for all examples on step length and the relative scaling between constraints and objective, it is still unclear whether the optimization parameters chosen in this paper are optimal. It is also unclear whether the current MMA parameters can conveniently adapt for extensions to

more complex materials. Therefore, it would be meaningful to develop a more general and easy-to-setup optimizer, likely taking advantage of second-order design sensitivity information.

7.2 Soft Hybrid Aerial Vehicle via Bistable Mechanism

7.2.1 Introduction

Hybrid aerial vehicles (HAVs) aim to improve flight efficiency and vehicle versatility by embedding in a single vehicle the ability to achieve multiple flight modes (Ke et al., 2018; Becker and Sheffler, 2016; Barbarino et al., 2011): a maneuverable copter mode capable of vertical takeoff and landing, hover, and other agile maneuvers; and a fuel-efficient fixed wing mode aimed at long-distance flight. Morphing aerial vehicles achieve different flight modes by changing the morphology of the vehicle itself (Falanga et al., 2018; Zhao et al., 2018a,b), but such morphing behavior often incurs additional costs of added weight, complexity and control (Xu et al., 2019a; Floreano et al., 2017; Tan and Chen, 2020; Morton and Papanikolopoulos, 2017).

In this paper, we propose a new HAV design (Figure 7.24), wherein the vehicle switches between a quadrotor mode and a fixed wing mode via a compliant bistable mechanism that deploys wings without requiring any additional actuators for reconfiguration beyond those included for normal flight. The design is inspired by (Bucki and Mueller, 2019), in which the arms of a quadrotor fold inward when the thrust is below a certain threshhold to allow the vehicle to fit through a tight space. We replace the folding mechanism with a bistable mechanism, allowing the HAV to remain in either mode in the absence of thrust. Our system replaces rigid quadrotor arms with soft material which deforms between two stable configurations, but constrains the propellers to a fixed ring. The first stable mode causes wings to deploy, and the other mode folds the wings. To transition between modes, we show that a sudden change in thrust can cause the battery to lag behind the motion of the outer ring due to its inertia, and that this inertial lag can be used to generate a mode switch without requiring any extra actuators. For this concept to work, the force required to

Figure 7.20: The fabricated HAV prototype with fixed wing (top) and quadrotor (bottom) modes.

actuate the bistable mechanism must fall within a specified range, based on the mass and thrust capabilties of the system.

Our vehicle is automatically optimized via *differentiable elasticity simulation*, where the mode switching mechanism is optimized by differentiating the material topology to achieve a compliant bistable mechanism, and the wing deployment mechanism is designed by differentiating the multi-bar connectivity and its reachability. Using topology optimization to design bistable structures has been studied for years. Most previous works rely on nonlinear finite element analysis to achieve large deformations, where robustly solving the displacement control and computing the analytic sensitivity information remain challenging. (Prasad and Diaz, 2005) maximized the distance between two equilibrium states, but they used a genetic algorithm to optimize the topology without sensitivity information, which has a low convergence rate. (James and Waisman, 2016) achieved bistability by manipulating the force-displacement curve directly. They minimized the backward switching force and set

a lower-bound for the forward switching force. Arc-length method was utilized to achieve displacement control. (Chen et al., 2018a) followed this formulation, but achieved displacement control by increasing a prescribed displacement gradually to a target one. (Chen et al., 2019) proposed a new formulation that optimizes the range of the force-displacement curve so that two-direction switching forces can both get optimized. We follow this idea as part of our formulation. We further model the displacement control as an energy minimization problem under equality constraints. With augmented Lagrangian method and projected Newton method (Teran et al., 2005), our approach is conceptually simple and practically robust even with large deformations.

We further utilize a multi-body mechanism for the wing design to enable large rotations. Our idea stems from (Swartz and James, 2019), which connects layers of meshes with clusters of springs with decayed stiffness to simulate pin joints between layers. The major advantage of this model is in the effective enforcement of positional continuity. Inspired by this work, we model joints as equality constraints between the interpolated displacements on the same material coordinate of two different layers in our optimization-based equilibrium solver. Consequently, each joint is placed inside a computational cell with its position optimized.

The contributions of this paper include:

- a novel soft Hybrid Aerial Vehicle (HAV) that leverages a compliant bistable mechanism for achieving two flight modes: 1) a quadrotor mode enabling maneuverability and 2) a fixed wing mode enabling efficient flight without requiring additional actuators;
- a topology optimization approach for both the bistable switching mechanism and the wing deployment mechanisms of the HAV; and
- experimental validation of the HAV design in a fabricated prototype.

7.2.2 Design and Validation of the HAV System

The automated design pipeline for our HAV system is materialized using efficient simulations of both elastostatics and elastodynamics. First, with a *differentiable* deformable body simulator

that solves for static force equilibrium, the design task is formulated as a smooth optimization problem where design variables are optimized with sensitivity information back-propagated from the objective (Section 7.2.2.2 and 7.2.2.3). Second, the elastodynamic simulation quickly validates each design and filters out inferior design choices before fabrication (Section 7.2.2.4).

The HAV system includes a central bistable structure and a foldable wing structure, wherein the state transition of the bistable structure folds wings inside or opens them up. These two parts are sequentially designed: a nonlinear elastic topology is first optimized to obtain bistability, and then the rotations of the joints of its arms act as boundary conditions for the wing design, where we reuse the differentiable equilibrium solver to optimize a multi-body rigid mechanism and obtain optimal joint locations for the wings.

7.2.2.1 Nonlinear Topology Optimization for Soft Materials

Topology optimization tackles the inverse simulation problem of finding a material distribution that fulfills mechanical and geometrical requirements under static equilibrium. We perform topology optimization on nonlinear hyperelastic materials to design the central bistable mechanism. Here we review the adopted topology optimization machinery.

Density-based topology optimization (Andreassen et al., 2011) usually represents a structure with a smooth density field $\rho \in [0, 1]$ in the material space Ω, where 1 represents fully solid and 0 represents fully void. Then topology optimization can often be desribed as a constrained optimization problem with the objective function L being the elastic potential, or compliance, and the constraints specifying static force equilibrium condition and the material volume target.

We choose the neo-Hookean hyperelasticity (Bonet and Wood, 1997) to model nonlinear elastic deformations that are crucial for bistable transitions. We then adopt a differentiable Material Point Method (MPM) (Jiang et al., 2016; Hu et al., 2019b) for discretization. MPM is a hybrid Lagrangian-Eulerian approach for computational solids, where the deformation is discretized on quadrature particles and physical equations are discretized on a grid. The sensitivity information is computed by differentiating the objective and the volume constraint

with respect to the design variables, which are a set of Lagrangian particles in this work (Li et al., 2021f). With gradients we solve the optimization problem using the Method of Moving Asymptotes (MMA) (Svanberg, 1987a).

7.2.2.2 Bistable Mechanism via Topology Optimization

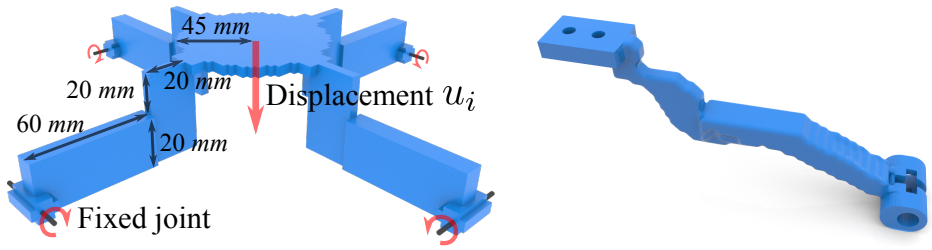
Bistable mechanism allows elastic structures to contain two stable equilibrium states, both of which can stably maintain their shapes without requiring any external loads. It is especially suitable for designing HAVs with two modes.

Following existing literature (James and Waisman, 2016; Chen et al., 2018a, 2019), we tackle bistable mechanism by controlling the force-displacement curve. The structure is bistable when the curve intersects the $f = 0$ (zero force) line in three distinct locations. The force-displacement curve is acquired through displacement control in the quasi-static setting: certain ports (Figure 7.21a) are constrained on a prescribed path by a given displacement sequence, then the reaction force on a port (the force needed to maintain the port on the track) is computed sequentially by static equilibrium on other nodes.

$$\min_{u} \; e(\rho, u) \quad \text{s.t.} \;\; u_i = u_i^* \tag{7.39}$$

where the displacement of node i is prescribed to some non-zero value u_i^*. Note that fixed nodes with zero displacements are eliminated from the degrees of freedom directly. The equality constraint is handled using the augmented Lagrangian method. With projected Newton and non-invertible line search (Li et al., 2021f; Nocedal and Wright, 1999), the solver remains robust under arbitrarily large displacement-control constraints. After the displacement field is acquired, the reaction force at the port i is then computed as $R_i = \frac{\partial e}{\partial u_i}$.

Similarly to (Chen et al., 2019), we maximize the difference of two switching forces (the forces required for snap-throughs between the two states), while minimizing the mean compliance under a force along the control path to guarantee sufficient structural stiffness.



(a) (b)

Figure 7.21: (a) The design configuration, where the design domain of each arm is of an L shape, and the control direction is visualized with the red arrows. Each arm can rotate around an axle on its end; (b) The final optimized arm.

The complete formulation of the bistable mechanism topology optimization is then

$$\begin{aligned} &\min_{\xi} \; n^T \left(\frac{\partial e(\rho, u^2)}{\partial u_i} - \frac{\partial e(\rho, u^1)}{\partial u_i} \right) + \alpha f^T u^3 \\ &\text{s.t.} \quad \begin{cases} u^1 = \operatorname{argmin}_u e(\rho, u) \;\; \text{s.t. } u_i = \bar{u}_i^1 \\ u^2 = \operatorname{argmin}_u e(\rho, u) \;\; \text{s.t. } u_i = \bar{u}_i^2 \\ u^3 = \operatorname{argmin}_u e(\rho, u) - u^T f \\ n^T \frac{\partial e(\rho, u^1)}{\partial u_i} \leq f_1^*, \quad n^T \frac{\partial e(\rho, u^2)}{\partial u_i} \geq -f_2^* \\ V(\rho) \leq \bar{V}, \end{cases} \end{aligned} \tag{7.40}$$

where n is the control path direction, f is the regularity force along n, and α controls the weighting between two objectives. $\bar{u}_i^1$ and $\bar{u}_i^2$ correspond to our expected peak and valley points in the force-displacement curve. f_1^* and f_2^* are target snap-through forces to match practical needs.

The derivative of the reaction force $R_i = \frac{\partial e}{\partial u_i}$ w.r.t design variable ξ contains term $\frac{d\widehat{u}_i}{d\rho}$, which can be acquired by differentiating the force equilibrium equation $\frac{\partial e}{\partial \widehat{u}_i} = 0$ on $\widehat{u}_i$ w.r.t ρ. This leads to $\frac{dR_i}{d\xi} = \left[\frac{d\rho}{d\xi}\right]^T \left(\frac{\partial e^2}{\partial \rho \partial u_i} + \frac{\partial^2 e}{\partial \rho \partial \widehat{u}_i} \left[\frac{\partial^2 e}{\partial \widehat{u}_i^2} \right]^{-1} \frac{\partial^2 e}{\partial \widehat{u}_i \partial u_i} \right)$.

The settings of the initial design domain, the control direction and the Dirichlet boundary condition are illustrated in Figure 7.21a. We model the initial topology following the configuration of a quadrotor and initialize the density in the design domain to be the target volume fraction everywhere. To provide clearance for the rotors, we use a 3D L shape domain,

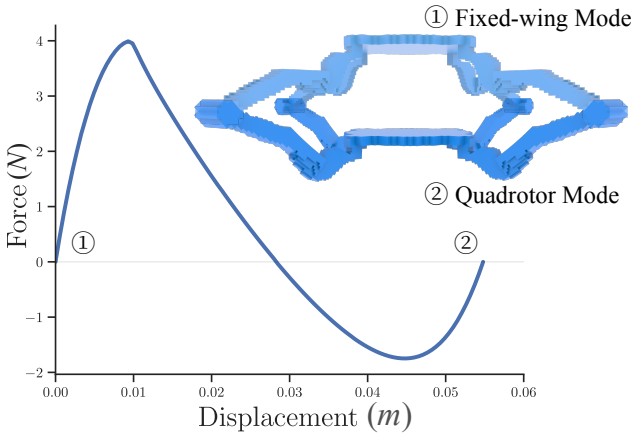


Figure 7.22: Force-displacement plot of the optimized topology with both equilibrium configurations visualized.

which is constructed by the constructive solid geometry difference between a box of 80 mm × 40 mm and a small box of 60 mm × 20 mm. The arm width is 5 mm for the first 20 mm close to the center and 10 mm for the remainder. Each such domain eventually becomes an arm connecting the outer ring with radius 125 mm to a central plate with radius 45 mm. The central plate represents the inner housing platform whose density is fixed to be 1 throughout the optimization. In addition to the displacement control constraints, the displacements of the axles (represented with small cylinders) are also fixed to be zero so that the arms can rotate around them (visualized with gray bars in Figure 7.21a). For efficiency, we only optimize over a quarter of the whole domain. Force equilibrium is solved under two symmetric boundary conditions so that the displacement field coincides with the one computed with the full domain. Likewise, to ensure identical arms, we set an extra symmetry constraint on the density field of each arm. The base Young's modulus and Poisson ratio of the arm are set as 5.5×10^6 Pa and 0.48 (TPU's material parameters). The Young's modulus of the central plate is set to be 1000 times larger (thus essentially treated as a rigid body). The expected peak and valley points of the force-displacement curve are chosen as $\bar{u}_i^1 = 10$ mm and $\bar{u}_i^2 = 50$ mm. Other parameters include target forces $f_1^* = 4$ N and $f_2^* = 1.2$ N, $f = -f_1^* n$, volume fraction $\bar{V} = 0.2$, and $\alpha = 50$. The force parameters are chosen according to the available mechanical parts of the quadrotor. The force magnitudes must be low enough to allow snap-through by the motors' thrust and high enough to prevent accidental snap-through due to gravity. The

optimized single arm is shown in Figure 7.21b. The full structure with four arms (Figure 7.22) demonstrates two equilibrium states.

7.2.2.3 Multi-Body Mechanism for the Wing Design

The folding of the wing is driven by the state transition of the bistable structure. One potential design strategy is to use a compliant mechanism (Pedersen et al., 2001). However, the rotation of the wing's leading edge is much larger than the arm of the bistable structure, and in this situation the compliant mechanism tends to generate low-density elements to act as joints, which makes the fabrication challenging. Inspired by (Swartz and James, 2019), we use multiple pieces of the continuum material to represent different components and use pinned joints (points in 2D and segments in 3D) to connect them. The material spaces of all pieces are aligned so that the same material coordinate refers to the same position in world coordinate at the undeformed state. We model a joint as one (for 2D joints) or a set of (for 3D joints) equality constraint(s) in the form of $u^1(X_i) = u^2(X_i)$ when solving the force equilibrium, where u^1 and u^2 are displacement fields of two different pieces that we connect and X_i is the common material coordinate. We only need a foldable wing skeleton, so we set the topology as the simplest form: we connect joint positions on the same component directly by straight bars. The only optimization variables are then the positions of these movable joints.

For any material coordinate X on some component i, its corresponding displacement can be written as $u(X) = A_i(X)u$, where $A_i(X)$ is the interpolation kernel for X on component i, and u is the concatenation of all nodal displacements from all components. So the above equality constraint for material coordinate X on components i and j can be written as a linear constraint $A_i(X)u - A_j(X)u = 0$. These constraints can be abstracted into a linear constraint $Hu = 0$. Likewise, since wing folding is driven by the transitioning of the bistable mechanism, its boundary condition is given by the rotation of the arm from the bistable

structure as a displacement control constraint . The equilibrium equation is then:

$$\min_{u} \; e(u) \quad \text{s.t.} \quad u_i = u^*, \quad H\widehat{u}_i = 0 \tag{7.41}$$

where we assume the displacement-controlled port is far from the joint so that matrix H is full-rank. This optimization problem can also be solved using augmented Lagrangian.

We use three components and two joints in the design, where each joint connects two components (Figure 7.23). This configuration choice is inspired by an earlier observation that three solid areas connected by two low-density joints will be formed when we only optimize over one single piece. One fixed joint is introduced to assist the rotational mechanism and another one serves as the rotation center of the arm. The formulation of the final wing optimization problem is then

$$\begin{aligned} &\min_{X_1, X_2} \; ||u_o^1 - \bar{u}_o^1||_2^2 + ||u_o^2 - \bar{u}_o^2||_2^2 + \alpha e(u^2) \\ \text{s.t.} \quad &||(X_1 + u^2(X_1)) - (X_2 + u^2(X_2))|| \geq s_1 \\ &X_2^x + (u^2)^x(X_2) \geq s_2, \quad X_1^y + (u^2)^y(X_1) \leq s_3 \end{aligned} \tag{7.42}$$

where X_1, X_2 are the two joints' material coordinates, o indices the wing tip, and $\alpha = 10$ controls the weighting between objectives. During forward motion, the transition from the quadrotor mode to the fixed-wing mode is achieved by pulling the central plate followed by releasing it (Section 7.2.2.4); it will experience a larger deformation (with displacement field u^2) than the deformation at the second equilibrium (with displacement field u^1). Therefore we need to control this more deformed state as well. Specifically, u^1 and u^2 are both solved using Equation (7.41) under the same joint displacement equality constraints but different displacement controls, $u_i^1 = \bar{u}_i^1$ and $u_i^2 = \bar{u}_i^2$. We also minimize the compliance to ensure that no energy is stored in the structure. The first constraint is used to sufficiently separate the two movable joints for easier fabrication. The other two constraints prevent joints from colliding with the bounding boxes of other parts.

The terms in the objective and the constraints can all be abstracted into $L(u, \mathbf{X})$ that

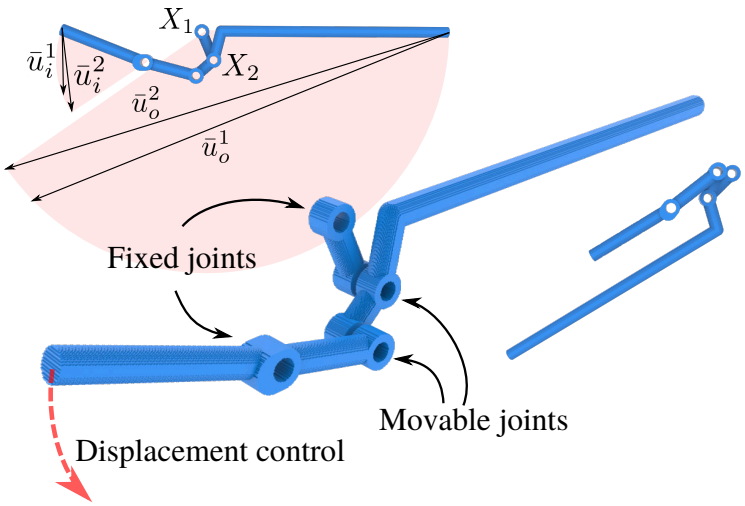


Figure 7.23: Optimized configuration. The deformation state of u^2 is illustrate on the bottom right corner. Top left shows how we choose the boundary condition and the objective.

depends on the mesh displacement and joint positions, where u also implicitly depends on $\mathbf{X}$ (the concatenation of the material coordinates of all movable joints to be optimized) through Equation (7.41). The derivative of L is given by $\frac{dL}{d\mathbf{X}} = \frac{\partial L}{\partial X} + \left[\frac{d\widehat{u_i}}{d\mathbf{X}}\right]^T \frac{\partial L}{\partial \widehat{u_i}}$, where $\frac{d\widehat{u_i}}{d\mathbf{X}}$ can be acquired by differentiating the equilibrium governing equation. The joint displacement constraints involve degrees of freedom that cannot be eliminated, thus we solve the governing equation using Lagrange multipliers and obtain $\frac{d\widehat{u_i}}{d\mathbf{X}}$, and then compute $\frac{dL}{dX}$ for solving the optimization problem in Equation (7.42) with MMA.

In Figure 7.23, the final optimized wing is shown, the design domain and boundary condition is illustrated at the top left corner and its deformed state is illustrated at the bottom right corner. Note that the optimization is done in 2D with point joints, but the mechanism works in the same way when the point joints are extruded to be segments. For the current prototype, we extrude by 8 mm to ensure that the 3D printed parts will be sturdy. The visualization is exactly how we assemble the wing parts in reality.

We simplify the optimized bistable mechanism arm as a straight bar and control the displacement on the tip to simulate rotation. The rotation angle (in radians) of the arm at the second equilibrium and the maximal-deformation state is 0.83 and 0.93, where $\bar{u}_i^1$ and $\bar{u}_i^2$ are set accordingly. The auxiliary fixed joint is at (35 mm, 20 mm) w.r.t. the fixed joint of the arm. The rotation of the wing tip is assumed to be around (55 mm, 20 mm). $\bar{u}_o^1$ and $\bar{u}_o^2$

are determined when the rotation angles are 2.45 and 2.46 respectively. The rotation of the wing does not need to be precise, since we optimize with L_2 norm. $\bar{u}_o^1$ and $\bar{u}_o^2$ only provide a guide for the optimization, and we can tune the rotation angles of the wing tip a little to adjust the position of the leading edge.

7.2.2.4 Validation through Forward Elastodynamic Simulations

The computational procedures described above for automatic designs are based on quasi-static approximations. To more reliably predict whether the designed system can function properly in practice, the whole system needs to be tested with dynamic forward simulations.

We use implicit MPM (Wang et al., 2020a) to perform the elastodynamic simulations. At each time step, we execute backward Euler time integration (taking step size Δt) with lagged Rayleigh damping (system matrix only) from the last time step, where the nonlinear system can be reformulated into an incremental potential minimization problem (Gast et al., 2015; Li et al., 2019a). Considering the assembly constraints, we solve

$$\begin{aligned} \min_{\Delta x} & \frac{1}{2}||\Delta x - \widetilde{\Delta x}||_M^2 + \frac{\gamma}{2\Delta t}||\Delta x||_{K^n}^2 + \Delta t^2(e(x^{n+1}) - \Delta x^T f_{ext}^n) \\ \text{s.t.} \quad & Hu = 0, \end{aligned}$$

where $K^n = \frac{\partial^2 e}{dx^2}(x^n)$, M is the mass matrix, e is the elastic potential, f_{ext} is the external force, γ is the damping coefficient, $v^{n,n+1}$ and $x^{n,n+1}$ denote velocities and positions from the known previous (n) and the unknown current $(n+1)$ time steps, $\widetilde{\Delta x} = v^n \Delta t + g\Delta t^2$, and $||x||_A^2$ represents $x^T A x$. The above optimization is solved by augmented Lagrangian with projected Newton and non-invertible line search (Wang et al., 2020a; Li et al., 2020). For stability we adopt a total Lagrangian formulation for tracking the deformation (de Vaucorbeil et al., 2020).

An assembled HAV is visualized in Figure 7.24. The electronics and the battery are represented with boxes to simulate their inertia effects only. There are four propellers which are simulated as four external forces applied on the corresponding positions. Mode transition of the HAV relies on the inertia of the central mass. The theoretical transition procedure is

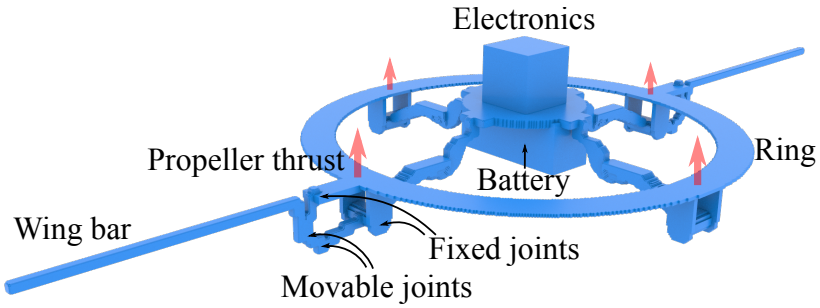


Figure 7.24: Final design (the simulated HAV system).

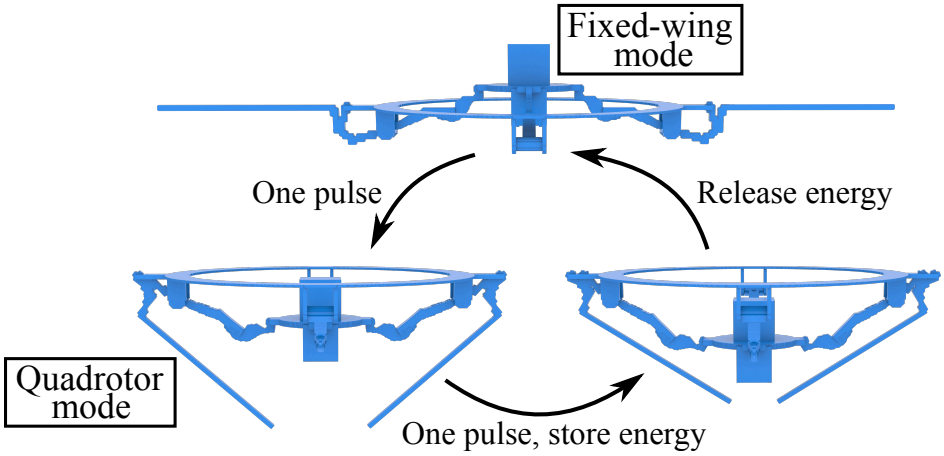


Figure 7.25: Theoretical state transition procedures.

illustrated in Figure 7.25. It is easy to transit from the fixed wing mode to the quadrotor mode (F to Q): when the ring is accelerated suddenly, the fictitious inertial force drags the central mass to the second equilibrium. However, it is not as straightforward to transit from the quadrotor mode to the fixed wing mode (Q to F). For propeller efficiency, we constrain our solution to only utilize thrust in the forward direction, which implies that the inertial force on the central mass is always pointing from the first equilibrium to the second equilibrium. Through experiments we discover a solution that utilizes the inertial force to temporarily store energy in the arm. After the propellers stop outputting forces, this energy is released to bounce the central mass from the second equilibrium towards the first equilibrium.

7.2.3 Fabrication and Evaluation

7.2.3.1 Prototype

We build the HAV prototype in Figure 7.20 using the electronics and propellers from an ARRIS X220 V2 5″ FPV Racing Drone (ARRISHobby, a). The ARRIS drone has four 5″ propellers placed 220 m apart along the diagonal, but we increase the diameter to 250 mm for our HAV to accommodate motion of the electronics housing. The combination of ARRIS X2206 2450kV brushless motors and Dalprop T5045C high efficiency propellers produces 8.9 N of thrust from each propeller when operating at 12 V (ARRISHobby, b). The drone is powered by a 4S 1500 mAh 100C LiPo battery. and can be remote controlled via radio signal. For experimental validation, we use the provided remote control and communicate over radio using the Radiolink AT9S. The total mass of the off-the-shelf system is 490 g including the battery. At 783 g, our prototype is not optimized for weight but the HAV has been shown in simulation and in testing to still have sufficient thrust.

For the switching mechanism, we 3D print four arms from flexible thermoplastic polyurethane (TPU) filament on a Makergear M3-ID 3D printer. Each arm replaces an arm of the ARRIS frame and connects from the housing to a ring laser cut out of 1/4″ thick acrylic sheet. To allow the arm to rotate and deform, it is attached to an cylindrical axel 20 mm below the ring. The motors mounted to the top of the ring actuate the propellers.

We 3D print the three components of the wing from the topology optimization out of polylactic acid (PLA) filament. The long leading edge of the wing mechanism is constructed from a 8 mm diameter carbon fiber rod. The second fixed joint is 3D printed as an extension to the connector holding the first fixed joint. We also add a rigid 3D printed bar extending into the ring that serves as an anchor for the wing surface.

The wing surface is a folded arc segmented into four panels and fabricated out of a 0.005″ thick polyethylene terephthalate (PET) film (Figure 7.20). Two thin wing ribs cut from 1/8″ thick PET glycol (PETG) rotate about X_2 and are attached to the second and third panels to guide the folding behavior (Figure 7.23). We determined that for proper folding action, the central fold needs to be biased towards the folded state, which we accomplish by

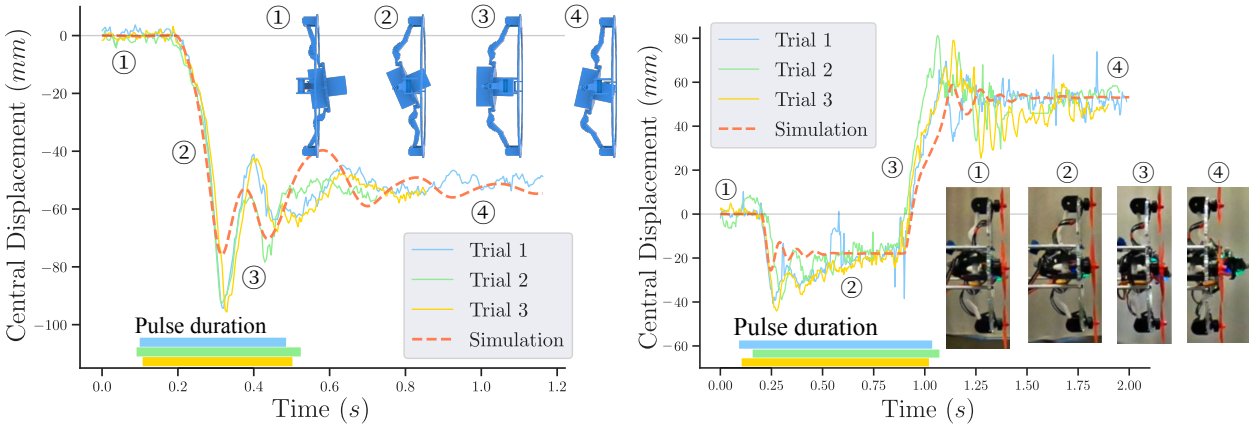


Figure 7.26: **(a)** Motion of the bistable mechanism during F to Q. **(b)** Motion of the bistable mechanism during Q to F.

sewing this fold.

The resulting prototype is bistable and able to support the weight of the battery against gravity when in fixed wing mode, as predicted by Figure 7.22, which will allow for a high climbing angle for the fixed wing. When the bistable mechanism snaps through, the wing surface is able to collapse and fold out as expected.

7.2.3.2 1D testing without wing

We mount the HAV to the end of a low friction boom with an arm length of 0.987 msuch that the thrust of the rotors is in the tangential direction. The setup was placed in a Vicon motion capture system to allow tracking of the boom's rotation. A GoPro Hero 8 mounted on the arm of the boom records the motion of the bistable mechanism.

We performed 3 trials of Q to F and 3 trials of F to Q. Each trial began with the HAV at rest. The F to Q transition consists of one short pulse with an average duration of 0.36s, while the Q to F transition consists of one long full thrust pulse with an average duration of 0.95s. Despite variability in control input due to manual operation of the HAV, the switching is reliable and the GoPro footage reveals that the central displacement of the bistable mechanism is similar to the predicted behavior of the simulation for both F to Q (Figure 7.26a) and Q to F (Figure 7.26b).

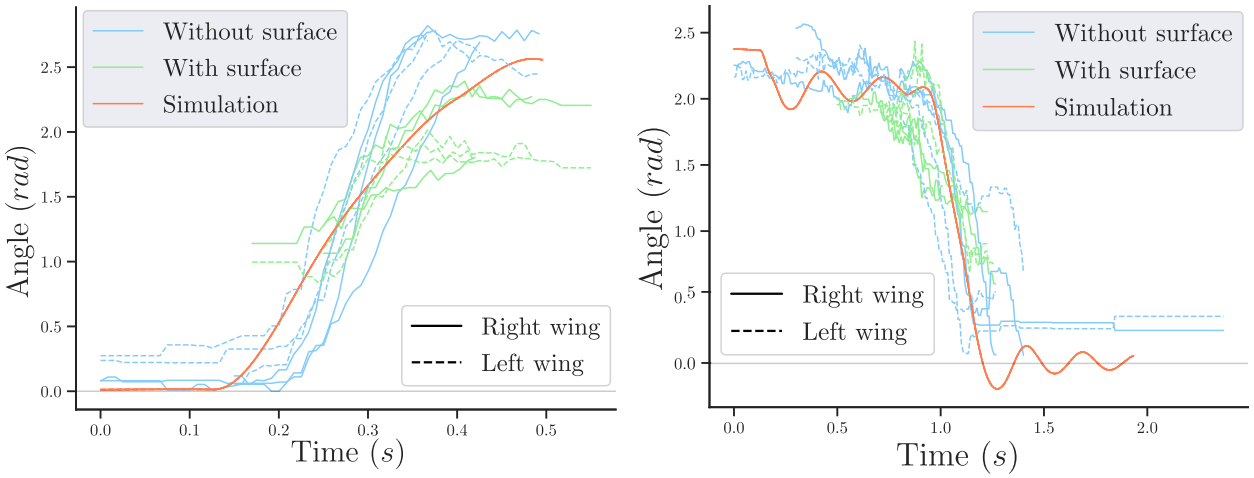


Figure 7.27: **(a)** F to Q 1D wing angle comparison. An angle of zero corresponds to the fixed wing mode. **(b)** Q to F 1D wing angle comparison.

7.2.3.3 1D testing with wing

When the HAV is constrained to rotate on a boom, the centripetal force dominates the rotation of the wing, causing both wing tips to rotate to the outside of the circle. Thus, performing boom tests with the wing did not provide useful information about the performance of the HAV. Instead, due to limited access to a large space and safety concerns with flying the untested prototype in a straight line, we conducted wing tests applying accelerations to the HAV prototype by hand. Grasping the ring of the HAV with the plane of the wing perpendicular to gravity, the experimenter manually exerted a pulsed force similar in profile to that measured during boom experiments. A GoPro fastened to the ceiling recorded the experiment at 120 fps, and the angle of the each wing was extracted. In all cases, the vehicle was able to successfully transition from Q to F and from F to Q, despite variability in the "control inputs." Compared to the simulation, where wing deployment was symmetric, imbalances in the frictional forces caused small differences in wing fold-out angle (Fig. 7.27a, 7.27b). When the wing surface was added, further deviation occurred since the non-optimized folding wings added extra resistance to the foldout mechanism and prevented the mechanism from reaching its extreme angles.

7.2.4 Discussion

We have demonstrated a novel design for a morphing HAV that leverages a bistable mechanism and vehicle accelerations to change flight modes. A topology optimization approach successfully generates a bistable mechanism with appropriate snap-through force that is low enough for the HAV's motors to trigger snap-through but high enough to prevent accidental mode switching. We showed that reliable mode switching is possible on the physically constructed system, even under variable acceleration inputs. We also demonstrate topology optimization of a folding wing mechanism driven by the motion of the bistable mechanism.

Additional design iterations will likely need to be performed to create an efficient HAV design. Sometimes during experiments, the fabricated prototype deforms at the connection of the TPU arm to the axel rather than rotating, and thus does not transmit the motion to the wing. We will iterate over the design of this connection so that there is no compliance at the axle. Redesigning the rigid HAV components to be cut from carbon fiber rather than 3D printed will allow for a reduction in system mass. Further, the wing surface has not yet been optimized.

Future work includes progressing to more 3D flight testing and the development of a controller for the switching maneuver. Since pitching motion is already required when switching between modes, we will leverage this for more efficient transitions. Pitching has been qualitatively observed to aid in both the Q to F and F to Q transitions.

CHAPTER 8

Conclusion

In this dissertation, we explored various tools for constructing physics-integrated digital twins. First, we developed diverse techniques to enhance the accuracy and efficiency of simulations. Additionally, we investigated methods that enable simulations based on reconstructions from real-world data. Finally, we demonstrated applications where mechanical designs can be created in a virtual environment and subsequently fabricated into real systems.

Our key contributions are outlined as follows:

Accurate Inelasticity Modeling. We introduce **ECI** (Section 3.1), a fully implicit inelasticity simulation framework. In this framework, we analytically derive plastic models with variational energies suitable for optimization-based time integration. Additionally, we propose **PlasticityNet** (Section 3.2) , a neural network-based extension of the ECI framework that generalizes to arbitrary plasticity models by learning plastic energies.

FEM-MPM Coupled Simulation. We present **BFEMP** (Section 4.1), a fully implicit strong FEM-MPM coupling framework utilizing IPC for interdomain contact. To improve efficiency and generality, we introduce **Dynamic Duo** (Section 4.2), which employs asynchronous time splitting to couple implicit FEM with explicit MPM.

Efficient Simulation. We develop **SPPD** (Section 5.1), a high-performance cloth simulation system that accelerates the convergence of projective dynamics by integrating subspace dynamics with Jacobian relaxations. Additionally, we propose **XPBI** (Section 5.2), a method that combines continuum plasticity theory with the fast XPBD simulator to enable efficient inelasticity simulations.

Real-to-Sim. By leveraging neural rendering and differentiable rendering techniques,

we enable several real-to-sim applications: we estimate physical parameters from videos (**PAC-NeRF**, Section 6.1), directly simulate real-world reconstructions (**PhysGaussian**, Section 6.2), create an interactive physics-based digital twin of a captured environment in virtual reality (**VR-GS**, Section 6.3), and generate simulation-ready garments from a casually captured single-view image (**Dress-1-to-3**, Section 6.4).

Sim-to-Real. We introduce **LETO** (Section 7.1), a novel topology optimization framework based on MPM. Using LETO, we successfully designed and fabricated an **HAV** (Section 7.2) with both fixed-wing and quadrotor modes.

APPENDIX A

Detailed Derivations of Variational Plasticity Models

A.1 Force-based Implicit Plasticity

Let's first derive the implicit plasticity treatment in a similar fashion to (Klár et al., 2016). Note that we will get a different expression which we claim is better. The idea is to not follow any variational principle and define implicite plasticity through implicit force balance. Note that in an updated Lagrangian MPM setting, we have

$$\boldsymbol{F}^{E,tr} = \left(\boldsymbol{I} + \sum_i \boldsymbol{v}_i^{n+1} \nabla w_{ip}^{\top} \right) \boldsymbol{F}^{E,n}. \tag{A.1}$$

A weak form derivation of the momentum balance equation

$$R(\boldsymbol{X}, 0)\boldsymbol{A}(\boldsymbol{X}, t) = \nabla^{\boldsymbol{X}} \cdot \boldsymbol{P} \tag{A.2}$$

would results in a "force" term (for node i)

$$\boldsymbol{f}_i = -\int_{\Omega_0} \boldsymbol{P} \nabla^{\boldsymbol{X}} Q_i d\boldsymbol{X}, \tag{A.3}$$

where $Q_i = Q_i(\boldsymbol{X}, t)$ is a test function. For MPM, we typically push forward Q_i onto its Eulerian Ω_n counterpart $q_i(\boldsymbol{x}, t)$ by observing from chain rule that $\nabla^{\boldsymbol{X}} Q_i = \boldsymbol{F}^{n\top} \nabla^{\boldsymbol{x}} q_i$. Further using the volume integral push forward property that $\int_{\Omega_t} g(\boldsymbol{x}) d\boldsymbol{x} = \int_{\Omega_0} G(\boldsymbol{X}) J(\boldsymbol{X}, t) d\boldsymbol{X}$, the push forward of (Equation (A.3)) is

$$\boldsymbol{f}_i = -\int_{\Omega_n} \frac{1}{J^n} \boldsymbol{P} \boldsymbol{F}^{n\top} \nabla^{\boldsymbol{x}} q_i d\boldsymbol{x}. \tag{A.4}$$

MPM uses particle quadratures to discretize the integral. Denoting $\nabla^{\boldsymbol{x}} q_i(\boldsymbol{x}_p^n, t^n)$ as ∇w_{ip}^n, we would then have

$$\boldsymbol{f}_i = -\sum_p \frac{V_p^n}{J_p^n} \boldsymbol{P}_p \boldsymbol{F}_p^{n\top} \nabla w_{ip}^n \tag{A.5}$$

$$= -\sum_p V_p^0 \boldsymbol{P}_p \boldsymbol{F}_p^{n\top} \nabla w_{ip}^n. \tag{A.6}$$

For implicit plasticity, we then plug in $\boldsymbol{P} = \frac{\partial \Psi^E}{\partial \boldsymbol{F}^E} \boldsymbol{F}^{P^{-\top}}$ (Bonet and Wood, 1997) to finally get

$$\boxed{\boldsymbol{f}_i^{n+1} = -\sum_p V_p^0 \frac{\partial \Psi^E}{\partial \boldsymbol{F}^E}(\boldsymbol{F}_p^{E,n+1}) \boldsymbol{F}_p^{P,n+1^{-\top}} \boldsymbol{F}_p^{n\top} \nabla w_{ip}^n,} \tag{A.7}$$

where for each particle p,

$$\boldsymbol{F}^{E,n+1} = \mathbf{Z}(\boldsymbol{F}^{E,tr}), \tag{A.8}$$

$$\boldsymbol{F}^{P,n+1} = \boldsymbol{F}^{E,n+1^{-1}} \boldsymbol{F}^{n+1} = \boldsymbol{F}^{E,n+1^{-1}} \boldsymbol{F}^{E,tr} \boldsymbol{F}^{P,n}. \tag{A.9}$$

Clearly, the dependency of $\boldsymbol{f}_i^{n+1}$ on $\boldsymbol{x}$ is in both the $\frac{\partial \Psi^E}{\partial \boldsymbol{F}^E}$ term and the $\boldsymbol{F}_p^{P,n+1^{-\top}}$ term. Note that by observing

$$\boldsymbol{F}^{n+1} = \boldsymbol{F}^{E,n+1} \boldsymbol{F}^{P,n+1} = \boldsymbol{F}^{E,tr} \boldsymbol{F}^{P,n} = \boldsymbol{F}^{E,tr} (\boldsymbol{F}^{E,n})^{-1} \boldsymbol{F}^n,$$

we know

$$\boldsymbol{F}^{P,n+1^{-\top}} = (\boldsymbol{F}^{E,n+1})^T (\boldsymbol{F}^{E,tr})^{-\top} (\boldsymbol{F}^{E,n})^T (\boldsymbol{F}^n)^{-\top},$$

so

$$\boldsymbol{F}^{P,n+1^{-\top}} \boldsymbol{F}^{n\top} = (\boldsymbol{F}^{E,n+1})^T (\boldsymbol{F}^{E,tr})^{-\top} (\boldsymbol{F}^{E,n})^T.$$

This means we don't actually need to track the plastic deformation gradient $\boldsymbol{F}^P$ on the

particles. We can use

$$\boxed{\boldsymbol{f}_i^{n+1} = -\sum_p V_p^0 \frac{\partial \Psi^E}{\partial \boldsymbol{F}^E}(\boldsymbol{F}_p^{E,n+1})(\boldsymbol{F}^{E,n+1})^T \boldsymbol{F}^{E,tr^{-\top}}(\boldsymbol{F}^{E,n})^T \nabla w_{ip}^n.} \tag{A.10}$$

Note that by replacing the future states $\boldsymbol{F}^{E,tr}$ and $\boldsymbol{F}^{E,n+1}$ by the last state $\boldsymbol{F}^{E,n}$, we get the a force formulation for explicit methods:

$$\boldsymbol{f}_i^{n+1} = -\sum_p V_p^0 \frac{\partial \Psi^E}{\partial \boldsymbol{F}^E}(\boldsymbol{F}_p^{E,n})(\boldsymbol{F}^{E,n})^T \nabla w_{ip}^n. \tag{A.11}$$

This is exactly the formulation we used in the traditional pipeline of MPM simulation for elastoplastic materials.

The formulation in (Klár et al., 2016) Instead of using (Equation (A.7)) or (Equation (A.10)), Klár et al. (2016) used an implicit force formulation derived from (Equation (A.11)), where only the stress term is made implicit:

$$\boldsymbol{f}_i^{n+1} = -\sum_p V_p^0 \frac{\partial \Psi^E}{\partial \boldsymbol{F}^E}(\boldsymbol{F}_p^{E,n+1}) \boldsymbol{F}_p^{E,n^\top} \nabla w_{ip}^n, \tag{A.12}$$

$$= -\sum_p V_p^0 \frac{\partial \Psi^E}{\partial \boldsymbol{F}^E}(\mathbf{Z}(\boldsymbol{F}_p^{E,tr})) \boldsymbol{F}_p^{E,n^\top} \nabla w_{ip}^n \tag{A.13}$$

which is equivalent to setting $\boldsymbol{F}_p^{P,n+1^{-\top}} \boldsymbol{F}_p^{n^\top} = \boldsymbol{F}_p^{E,n^\top}$, i.e., $\boldsymbol{F}_p^{P,n} = \boldsymbol{F}_p^{P,n+1}$ in (Equation (A.7)). Thus (Klár et al., 2016)'s implicit force-plasticity can be treated as a particular semi-implicit choice of modifying (Equation (A.7)).

Let $\mathbf{B}(\boldsymbol{\Sigma}) = \frac{\partial \Psi^E}{\partial \boldsymbol{\Sigma}^E}(\mathbf{Z}(\boldsymbol{\Sigma}))$. The hessian of $\mathbf{B}$ is

$$\nabla \mathbf{B} = (\frac{d\mathbf{Z}}{d\boldsymbol{\Sigma}})^\top \frac{\partial^2 \Psi^E}{\partial (\boldsymbol{\Sigma}^E)^2}(\mathbf{Z}) \tag{A.14}$$

where $\frac{d\mathbf{Z}}{d\boldsymbol{\Sigma}}$ is not symmetric. So $\mathbf{B}$ cannot be a gradient of a smooth energy.

Remark: Regardless of whether (Equation (A.7)) or (Equation (A.13)) is adopted, we could easily verify that in the plastic region,

$$\boldsymbol{f}_i \neq -\frac{\partial E}{\partial \boldsymbol{x}_i}$$

where

$$E = \int_{\Omega_0} \Psi^E(\boldsymbol{F}^E) d\boldsymbol{X} \approx \sum_p V_p^0 \Psi^E(\boldsymbol{F}^E) = \sum_p V_p^0 \Psi^E(\mathbf{Z}(\boldsymbol{F}^{E,tr})). \tag{A.15}$$

This is clear because in the region outside (but not touching) the yield surface (consider a 1D metal), (Equation (A.15)) is a constant with zero derivative. But clearly at this configuration one should not get a zero force.

A.2 Augmenting the Energy: A One-Dimensional Inspiration

Consider a one-dimensional elastoplastic spring with its left end fixed at $x = 0$. Let its rest length be $V_0 = 1$. Under the framework of finite strain elastoplasticity $F = F^E F^P$, we have

$$F(x) = x. \tag{A.16}$$

Looking at $t^n \to t^{n+1}$, we get

$$x^{n+1} = F^{n+1} = F^{E,n+1} F^{P,n+1} = F^{E,tr} F^{P,n}, \tag{A.17}$$

where $F^{E,tr}$ is a trial stress by assuming purely elastic deformation:

$$F^{E,tr}(x) = \frac{1}{F^{P,n}} x. \tag{A.18}$$

We can use hyperelasticity to define the elastic response:

$$\Psi^E(F^E) = \frac{k}{2} \log(F^E)^2. \tag{A.19}$$

We'll pick stiffness $k = 1$ for brevity. The Kirchhoff stress can then be given by

$$\tau(F^E) := \frac{\partial \Psi^E}{\partial F^E} F^{E^\top} = \log(F^E). \tag{A.20}$$

Let the yield stress be $\tau_Y = \log(F_Y)$, where $F_Y \in [1, \infty)$ is a critical strain, and define the yield function to be $\tau - \tau_Y \leq 0$, we can then follow standard plasticity treatment (Simo and Hughes, 1998) to reach a simple return mapping procedure with the form

$$F^{E,n+1} = Z(F^{E,tr}) = \begin{cases} F^{E,tr} & F^{E,tr} \leq F_Y \\ F_Y & \text{otherwise} \end{cases}. \tag{A.21}$$

In terms of the Hencky strain $\epsilon^E = \log(\boldsymbol{F}^E)$, return mapping comes from a discretized plastic flow rule with the form (assuming $F^{E,tr} > F_Y$)

$$\epsilon^{E,tr} - \epsilon^{E,n+1} = \delta\gamma \tag{A.22}$$

$$\Leftrightarrow \quad \delta\gamma = \log\left(\frac{F^{E,tr}}{F^{E,n+1}}\right) = \log\left(\frac{F^{E,tr}}{F_Y}\right) > 0. \tag{A.23}$$

Theorem A.2.1 (Augmented energy density for springs). In the problem setting described above ($V_0 = 1$, $k = 1$), using the following energy density function outside the yield surface

$$\boxed{\Psi(x) = \begin{cases} \Psi^E(Z(F^{E,tr}(x))) + \tau_Y \delta\gamma(F^{E,tr}(x)) & F^{E,tr} > F_Y \\ \Psi^E(F^{E,tr}(x)) & \text{otherwise} \end{cases}} \tag{A.24}$$

reveals a force that is equivalent to what one would get if one performed the force-based implicit plasticity.

Proof. The result for the elastic region is trivial and we skip the proof.

Note that we only have one element with volume $V^0 = 1$, thus the total energy $E = \Psi$.

For x in the plastic region ($F^{E,tr} > F_Y$), since $Z(F^{E,tr}(x)) = F_Y$ is a constant, the derivative of the first term in the energy vanishes. Plugging in (Equation (A.18)) and

(Equation (A.23)) for $F^{E,tr}$ and $\delta\gamma$, we get

$$-f = \frac{\partial E}{\partial x} = \frac{\partial \Psi}{\partial x} = \tau_Y \frac{\partial \delta\gamma}{\partial x} = \tau_Y \frac{F_Y}{F^{E,tr}} \frac{1}{F_Y} \frac{1}{F^{P,n}} = \frac{\tau_Y}{x}. \tag{A.25}$$

Next we take a force-based implicit plasticity approach to derive the force. We could discretize the problem with linear FEM by using a material space interpolation function $N_i(X) = X, X \in [0, 1]$. Therefore $\nabla^X N = 1$. Since we have only one degree of freedom and one element, we get

$$-f = \int_{\Omega_0} P \nabla^X N dX = P = \frac{\partial \Psi^E}{\partial F^E}(Z(F^{E,tr}))F^{P^{-1}} = \frac{\log F_Y}{F_Y} F^{P^{-1}}. \tag{A.26}$$

Since

$$x = F^{n+1} = F^{E,n+1} F^{P,n+1} = F_Y F^{P,n+1} \quad \Rightarrow \quad F^{P^{-1}} = \frac{F_Y}{x},$$

we have

$$-f = \frac{\log F_Y}{F_Y} \frac{F_Y}{x} = \frac{\tau_Y}{x} \tag{A.27}$$

Therefore, two approaches result in the same force on this one-DOF system. □

To enable a line search projected Newton's method applied on $\Psi(x)$ defined by (Equation (A.24)), we further prove

Theorem A.2.2. $\Psi(x)$ defined by (Equation (A.24)) is piecewise C^∞ and everywhere C^1.

Proof. Piecewise high order smoothness is easy to show so we skip. Below we only prove that the energy density function has a continuous first-order derivative at $F^{E,tr} = F_Y$.

When $F^{E,tr} \leq F_Y$, $\Psi = \Psi_E(F^{E,tr}(x))$,

$$\frac{\partial \Psi}{\partial F^{E,tr}} = \frac{\partial \Psi^E}{\partial F^E}(F^{E,tr}) = \frac{\log F^{E,tr}}{F^{E,tr}}, \tag{A.28}$$

and at $F^{E,tr} = F_Y$, it evaluates to $\frac{\log F_Y}{F_Y}$.

When $F^{E,tr} > F_Y$, $\Psi = \Psi^E(Z(F^{E,tr}(x))) + \tau_Y \delta\gamma(F^{E,tr}(x))$. Since $Z(F^{E,tr}) = F_Y$ is a constant, the first term has derivative 0. Differentiating the second term gives

$$\frac{\partial \Psi}{\partial F^{E,tr}} = \tau_Y \frac{\partial \delta\gamma}{\partial F^{E,tr}} = \tau_Y \frac{F_Y}{F^{E,tr}} \frac{1}{F_Y} = \frac{\tau_Y}{F^{E,tr}}, \tag{A.29}$$

and at $F^{E,tr} = F_Y$, it also evaluates to $\frac{\log F_Y}{F_Y}$.

□

A.3 Extending to Multiple Dimensions: Von-Mises

A.3.1 Augmented Energy Theorem

It is equivalent and simpler to discuss in the principle stretch space. The deformation gradient is decomposed as $\boldsymbol{F} = \boldsymbol{U}\,\mathrm{diag}(\boldsymbol{\Sigma})\boldsymbol{V}^\top$. The return mapping is then defined as $\mathbf{Z}(\boldsymbol{F}) = \boldsymbol{U}\,\mathrm{diag}(\hat{\mathbf{Z}}(\boldsymbol{\Sigma}))\boldsymbol{V}^\top$.

Let's define:

$$\boldsymbol{\epsilon} = \ln(\boldsymbol{\Sigma}) \quad \hat{\boldsymbol{\epsilon}} = \boldsymbol{\epsilon} - \frac{\mathrm{sum}(\boldsymbol{\epsilon})}{d}\mathbf{1} \quad \delta\gamma = \|\hat{\boldsymbol{\epsilon}}\| - \frac{\tau_Y}{2\mu} \tag{A.30}$$

The return mapping for the von-Mises plasticity model in the principal stretch space is defined as

$$\boldsymbol{Z}(\boldsymbol{\Sigma}) = \begin{cases} \boldsymbol{\Sigma}, & \delta\gamma \le 0 \\ \exp\left(\boldsymbol{\epsilon} - \delta\gamma \frac{\hat{\boldsymbol{\epsilon}}}{\|\hat{\boldsymbol{\epsilon}}\|}\right), & otherwise \end{cases} \tag{A.31}$$

Denote $\mathbf{H} = \boldsymbol{\epsilon} - \delta\gamma \frac{\hat{\boldsymbol{\epsilon}}}{\|\hat{\boldsymbol{\epsilon}}\|}$, then

$$\frac{d\mathbf{H}}{d\boldsymbol{\Sigma}} = \mathrm{diag}(\boldsymbol{\Sigma}^{-1}) - (1 - \frac{\tau_Y}{2\mu\|\hat{\boldsymbol{\epsilon}}\|})(\mathrm{diag}(\boldsymbol{\Sigma}^{-1}) - \frac{1}{d}\mathbf{1}\boldsymbol{\Sigma}^{-\top}) - \frac{\tau_Y}{2\mu\|\hat{\boldsymbol{\epsilon}}\|}\hat{\boldsymbol{s}}\hat{\boldsymbol{s}}^\top \mathrm{diag}(\boldsymbol{\Sigma}^{-1}) \tag{A.32}$$

where $\hat{\boldsymbol{s}} = \frac{\hat{\epsilon}}{\|\hat{\boldsymbol{\epsilon}}\|}$.

The Jacobian of return mapping outside the yeild surface is

$$\frac{d\boldsymbol{Z}}{d\boldsymbol{\Sigma}} = \mathrm{diag}(\boldsymbol{Z})\frac{d\mathbf{H}}{d\boldsymbol{\Sigma}} \tag{A.33}$$

Define the **augmented elastoplastic energy density function** as:

$$\boxed{\Psi(\boldsymbol{F}) = \begin{cases} \Psi^E(\boldsymbol{F}), & \delta\gamma(\boldsymbol{F}) \leq 0 \\ \Psi^E(Z(\boldsymbol{F})) + \tau_Y \delta\gamma(\boldsymbol{F}), & otherwise \end{cases}} \tag{A.34}$$

Lemma A.3.1.

$$\boxed{\forall \boldsymbol{F}, \boldsymbol{\tau}(\boldsymbol{Z}(\boldsymbol{F})) \frac{\partial \Psi^E}{\partial \boldsymbol{F}^E}(\boldsymbol{Z}(\boldsymbol{F})) \boldsymbol{Z}(\boldsymbol{F})^\top \equiv \frac{\partial \Psi(\boldsymbol{F})}{\partial \boldsymbol{F}} \boldsymbol{F}^T} \tag{A.35}$$

Proof. In the principal stretch space, it is equivalent to prove that

$$\forall \boldsymbol{\Sigma}, \frac{\partial \Psi^E}{\partial \boldsymbol{\Sigma}^E}(\boldsymbol{Z}(\boldsymbol{\Sigma})) \odot \boldsymbol{Z}(\boldsymbol{\Sigma}) \equiv \frac{\partial \Psi(\boldsymbol{\Sigma})}{\partial \boldsymbol{\Sigma}} \odot \boldsymbol{\Sigma},$$

where $\odot$ denote the element-wise multiplication between two vectors.

It is trivial that inside the yield surface, the statement is true, since that $\boldsymbol{Z}$ is the identity map.

Outside the yield surface, we have

$$\begin{aligned} \frac{\partial \Psi^E}{\partial \boldsymbol{\Sigma}^E}(\boldsymbol{Z}) \odot \boldsymbol{Z} =& 2\mu \ln(\boldsymbol{Z}) + \lambda \operatorname{sum}(\ln \boldsymbol{Z}) \mathbf{1} \\ =& 2\mu \mathbf{H} + \lambda \operatorname{sum}(\mathbf{H}) \mathbf{1} \end{aligned} \tag{A.36}$$

$$\begin{aligned} \frac{\partial \Psi(\boldsymbol{\Sigma})}{\partial \boldsymbol{\Sigma}} =& (\frac{d\boldsymbol{Z}}{d\boldsymbol{\Sigma}})^\top \frac{\partial \Psi^E}{\partial \boldsymbol{\Sigma}^E}(\boldsymbol{Z}) + \tau_Y (\frac{d\hat{\boldsymbol{\epsilon}}}{d\boldsymbol{\Sigma}})^\top \hat{\boldsymbol{s}} \\ =& (\frac{d\boldsymbol{Z}}{d\boldsymbol{\Sigma}})^\top \left(2\mu \mathbf{H} \odot \boldsymbol{Z}^{-1} + \lambda \operatorname{sum}(\mathbf{H}) \boldsymbol{Z}^{-1}\right) + \tau_Y (\operatorname{diag}(\boldsymbol{\Sigma}^{-1}) - \frac{1}{d} \mathbf{1} \boldsymbol{\Sigma}^{-\top})^\top \hat{\boldsymbol{s}} \\ =& (\frac{d\boldsymbol{Z}}{d\boldsymbol{\Sigma}})^\top \left(2\mu \mathbf{H} \odot \boldsymbol{Z}^{-1} + \lambda \operatorname{sum}(\mathbf{H}) \boldsymbol{Z}^{-1}\right) + \tau_Y \hat{\boldsymbol{s}} \odot \boldsymbol{\Sigma}^{-1} - \frac{\operatorname{sum}(\hat{\boldsymbol{s}})}{d} \boldsymbol{\Sigma}^{-1} \\ =& (\frac{d\mathbf{H}}{d\boldsymbol{\Sigma}})^\top (2\mu \mathbf{H} + \lambda \ \operatorname{sum}(\mathbf{H}) \mathbf{1}) + \tau_Y \hat{\boldsymbol{s}} \odot \boldsymbol{\Sigma}^{-1} \end{aligned} \tag{A.37}$$

$$
\begin{aligned}
2\mu(\frac{d\mathbf{H}}{d\boldsymbol{\Sigma}})^\top\mathbf{H} =& 2\mu\mathbf{H}\odot\boldsymbol{\Sigma}^{-1} - (2\mu - \frac{\tau_Y}{\|\hat{\boldsymbol{\epsilon}}\|})(\mathbf{H}\odot\boldsymbol{\Sigma}^{-1} - \frac{\mathrm{sum}(\mathbf{H})}{d}\boldsymbol{\Sigma}^{-1}) - \frac{\tau_Y}{\|\hat{\boldsymbol{\epsilon}}\|}\hat{\boldsymbol{s}}^\top\mathbf{H}(\hat{\boldsymbol{s}}\odot\boldsymbol{\Sigma}^{-1}) \\
=& 2\mu\mathbf{H}\odot\boldsymbol{\Sigma}^{-1} - (2\mu - \frac{\tau_Y}{\|\hat{\boldsymbol{\epsilon}}\|})(\mathbf{H}\odot\boldsymbol{\Sigma}^{-1} - \frac{\mathrm{sum}(\mathbf{H})}{d}\boldsymbol{\Sigma}^{-1}) - \frac{\tau_Y}{\|\hat{\boldsymbol{\epsilon}}\|}(\hat{\boldsymbol{s}}^\top\boldsymbol{\epsilon} - \delta\gamma)(\hat{\boldsymbol{s}}\odot\boldsymbol{\Sigma}^{-1}) \\
=& 2\mu\mathbf{H}\odot\boldsymbol{\Sigma}^{-1} - (2\mu - \frac{\tau_Y}{\|\hat{\boldsymbol{\epsilon}}\|})(\mathbf{H} - \frac{\mathrm{sum}(\mathbf{H})}{d}\mathbf{1})\odot\boldsymbol{\Sigma}^{-1} - \frac{\tau_Y}{\|\hat{\boldsymbol{\epsilon}}\|}(\hat{\boldsymbol{s}}^\top\hat{\boldsymbol{\epsilon}} - \delta\gamma)(\hat{\boldsymbol{s}}\odot\boldsymbol{\Sigma}^{-1}) \\
=& 2\mu\mathbf{H}\odot\boldsymbol{\Sigma}^{-1} - (2\mu - \frac{\tau_Y}{\|\hat{\boldsymbol{\epsilon}}\|})(\mathbf{H} - \frac{\mathrm{sum}(\mathbf{H})}{d}\mathbf{1})\odot\boldsymbol{\Sigma}^{-1} - \frac{\tau_Y}{\|\hat{\boldsymbol{\epsilon}}\|}(\|\hat{\boldsymbol{\epsilon}}\| - \delta\gamma)(\hat{\boldsymbol{s}}\odot\boldsymbol{\Sigma}^{-1}) \\
=& 2\mu\mathbf{H}\odot\boldsymbol{\Sigma}^{-1} - \tau_Y\hat{\boldsymbol{s}}\odot\boldsymbol{\Sigma}^{-1} \\
& - \frac{2\mu}{\|\hat{\boldsymbol{\epsilon}}\|}(\|\hat{\boldsymbol{\epsilon}}\| - \frac{\tau_Y}{2\mu})(\mathbf{H} - \frac{\mathrm{sum}(\mathbf{H})}{d}\mathbf{1})\odot\boldsymbol{\Sigma}^{-1} + \frac{\tau_Y\delta\gamma}{\|\hat{\boldsymbol{\epsilon}}\|}(\hat{\boldsymbol{s}}\odot\boldsymbol{\Sigma}^{-1}) \\
=& 2\mu\mathbf{H}\odot\boldsymbol{\Sigma}^{-1} - \tau_Y\hat{\boldsymbol{s}}\odot\boldsymbol{\Sigma}^{-1} - \frac{\delta\gamma}{\|\hat{\boldsymbol{\epsilon}}\|}(2\mu(\hat{\boldsymbol{\epsilon}} - \delta\gamma\hat{\boldsymbol{s}}))\odot\boldsymbol{\Sigma}^{-1} + \frac{\tau_Y\delta\gamma}{\|\hat{\boldsymbol{\epsilon}}\|}(\hat{\boldsymbol{s}}\odot\boldsymbol{\Sigma}^{-1}) \\
=& 2\mu\mathbf{H}\odot\boldsymbol{\Sigma}^{-1} - \tau_Y\hat{\boldsymbol{s}}\odot\boldsymbol{\Sigma}^{-1} - \frac{\delta\gamma}{\|\hat{\boldsymbol{\epsilon}}\|}(2\mu(\|\hat{\boldsymbol{\epsilon}}\| - \delta\gamma))\hat{\boldsymbol{s}}\odot\boldsymbol{\Sigma}^{-1} + \frac{\tau_Y\delta\gamma}{\|\hat{\boldsymbol{\epsilon}}\|}(\hat{\boldsymbol{s}}\odot\boldsymbol{\Sigma}^{-1}) \\
=& 2\mu\mathbf{H}\odot\boldsymbol{\Sigma}^{-1} - \tau_Y\hat{\boldsymbol{s}}\odot\boldsymbol{\Sigma}^{-1}
\end{aligned}
\tag{A.38}
$$

$$
\begin{aligned}
(\frac{d\mathbf{H}}{d\boldsymbol{\Sigma}})^\top\mathbf{1} =& \boldsymbol{\Sigma}^{-1} - (1 - \frac{\tau_Y}{2\mu\|\hat{\boldsymbol{\epsilon}}\|})(\boldsymbol{\Sigma}^{-1} - \boldsymbol{\Sigma}^{-1}) - \frac{\tau_Y}{2\mu\|\hat{\boldsymbol{\epsilon}}\|}\,\mathrm{sum}(\hat{\boldsymbol{s}})\,\mathrm{diag}(\boldsymbol{\Sigma}^{-1})\hat{\boldsymbol{s}} \\
=& \boldsymbol{\Sigma}^{-1}
\end{aligned}
\tag{A.39}
$$

So we finally get

$$
\frac{\partial\Psi(\boldsymbol{\Sigma})}{\partial\boldsymbol{\Sigma}}\odot\boldsymbol{\Sigma} = (2\mu\mathbf{H}\odot\boldsymbol{\Sigma}^{-1} + \lambda\,\mathrm{sum}(\mathbf{H})\boldsymbol{\Sigma}^{-1})\odot\boldsymbol{\Sigma} = 2\mu\mathbf{H} + \lambda\,\mathrm{sum}(\mathbf{H})\mathbf{1} = \frac{\partial\Psi^E}{\partial\boldsymbol{\Sigma}^E}(\boldsymbol{Z})\odot\boldsymbol{Z} \tag{A.40}
$$

□

Theorem A.3.2 (Augmented energy theorem for von-Mises plasticity). The energy density function Equation (A.34) reveals a force that is equivalent to what one would get if one performed the force-based implicit plasticity.

Proof. According to Lemma Lemma A.3.1 and Eq Equation (A.10),

$$\begin{aligned}
\boldsymbol{f}_i^{n+1} &= -\sum_p V_p^0 \frac{\partial \Psi^E}{\partial \boldsymbol{F}^E}(\boldsymbol{F}_p^{E,n+1})(\boldsymbol{F}^{E,n+1})^T \boldsymbol{F}^{E,tr^{-\top}}(\boldsymbol{F}^{E,n})^T \nabla w_{ip}^n \quad \text{(implicit plasticity force)} \\
&= -\sum_p V_p^0 \frac{\partial \Psi^E}{\partial \boldsymbol{F}^E}(\boldsymbol{Z}(\boldsymbol{F}_p^{E,tr}))(\boldsymbol{Z}(\boldsymbol{F}_p^{E,tr}))^T \boldsymbol{F}^{E,tr^{-\top}}(\boldsymbol{F}^{E,n})^T \nabla w_{ip}^n \\
&= -\sum_p V_p^0 \frac{\partial \Psi}{\partial \boldsymbol{F}}(\boldsymbol{F}_p^{E,tr}) \boldsymbol{F}_p^{E,n^\top} \nabla w_{ip}^n \\
&\qquad \text{(implicit MPM force from the augmented elastoplastic energy)}
\end{aligned} \tag{A.41}$$

□

Von-Mises Plasticity under Linear Hardening Under linear hardening, the Kirchhoff stress in principle stretch space is

$$\boldsymbol{\tau}(\boldsymbol{Z}(\boldsymbol{\Sigma})) = \begin{cases} 2\mu \operatorname{dev} \boldsymbol{\epsilon} + (\lambda + \frac{2\mu}{d}) \log J \mathbf{1}, & \delta\gamma \le 0, \\ \tau_Y^n \frac{\operatorname{dev} \boldsymbol{\epsilon}}{\|\operatorname{dev} \boldsymbol{\epsilon}\|} + 2\mu K \delta\gamma \frac{\operatorname{dev} \boldsymbol{\epsilon}}{\|\operatorname{dev} \boldsymbol{\epsilon}\|} + (\lambda + \frac{2\mu}{d}) \log J \mathbf{1}, & \text{otherwise}, \end{cases} \tag{A.42}$$

where $K \in [0, 1]$ is the linear hardening coefficient. Here τ_Y is not constant over time, instead, it evolves discretely by the linear hardening rule:

$$\tau_Y^{n+1} = \tau_Y^n + 2\mu K \delta\gamma. \tag{A.43}$$

We can rearrange the stress terms into

$$\boldsymbol{\tau}(\boldsymbol{Z}(\boldsymbol{\Sigma})) = \begin{cases} 2\mu \operatorname{dev} \boldsymbol{\epsilon} + (\lambda + \frac{2\mu}{d}) \log J \mathbf{1}, & \delta\gamma \le 0, \\ K(2\mu \operatorname{dev} \boldsymbol{\epsilon} + (\lambda + \frac{2\mu}{d}) \log J \mathbf{1}) + (1 - K)(\tau_Y^n \frac{\operatorname{dev} \boldsymbol{\epsilon}}{\|\operatorname{dev} \boldsymbol{\epsilon}\|} \mathbf{1} + (\lambda + \frac{2\mu}{d}) \log J \mathbf{1}), & \text{otherwise}. \end{cases} \tag{A.44}$$

This leads to the following augmented energy:

$$\Psi(\boldsymbol{F}) = \begin{cases} \Psi^E(\boldsymbol{F}) & \delta\gamma \leq 0 \\ K\Psi^E(\boldsymbol{F}) + (1-K)\left(\Psi^E(\boldsymbol{Z}(\boldsymbol{F})) + \tau_Y^n \delta\gamma\right) & \text{otherwise} \end{cases} \tag{A.45}$$

The energy is a linear combination of the purely elastic case ($K = 0$) the perfectly plastic case($K = 1$).

A.3.2 Implementation Tricks

By Theorem Theorem A.3.2, the gradient of the augmented energy in the principal stretch space can be simplified as

$$\frac{\partial \Psi}{\partial \boldsymbol{\Sigma}} = \frac{\partial \Psi^E}{\partial \boldsymbol{\Sigma}^E}(\boldsymbol{Z}(\boldsymbol{\Sigma})) \odot \boldsymbol{Z}(\boldsymbol{\Sigma}) \odot \boldsymbol{\Sigma}^{-1} = (2\mu\mathbf{H} + \lambda\,\mathrm{sum}(\mathbf{H})\mathbf{1}) \odot \boldsymbol{\Sigma}^{-1}, \tag{A.46}$$

which avoids the evaluation of $\frac{d\mathbf{Z}}{d\boldsymbol{\Sigma}}$.

Then the hessian is

$$\frac{\partial^2 \Psi}{\partial \boldsymbol{\Sigma}^2} = \mathrm{diag}(\boldsymbol{\Sigma}^{-1})(2\mu\boldsymbol{I} + \lambda\mathbf{1}\mathbf{1}^\top)\frac{d\mathbf{H}}{d\boldsymbol{\Sigma}} - \mathrm{diag}(2\mu\mathbf{H} + \lambda\,\mathrm{sum}(\mathbf{H})\mathbf{1})\,\mathrm{diag}(\boldsymbol{\Sigma}^{-2}) \tag{A.47}$$

Following (Jiang et al., 2016), the other coefficients needed to evaluate hessian w.r.t. deformation gradient are

$$\frac{\partial_{\boldsymbol{\Sigma}^i}\Psi - \partial_{\boldsymbol{\Sigma}^j}\Psi}{\boldsymbol{\Sigma}^i - \boldsymbol{\Sigma}^j} = \frac{1}{\boldsymbol{\Sigma}^i\boldsymbol{\Sigma}^j}(-2\mu\mathbf{H}^i + 2\mu\boldsymbol{\Sigma}^i\frac{\ln(\boldsymbol{Z}^i) - \ln(\boldsymbol{Z}^j)}{\boldsymbol{Z}^i - \boldsymbol{Z}^j}\frac{\boldsymbol{Z}^i - \boldsymbol{Z}^j}{\boldsymbol{\Sigma}^i - \boldsymbol{\Sigma}^j} - \lambda\,\mathrm{sum}(\mathbf{H})) \tag{A.48}$$

where

$$\frac{\boldsymbol{Z}^i - \boldsymbol{Z}^j}{\boldsymbol{\Sigma}^i - \boldsymbol{\Sigma}^j} = (1 - \frac{\delta\gamma}{\|\hat{\boldsymbol{\epsilon}}\|})(\frac{\exp(\mathbf{H}^i) - \exp(\mathbf{H}^j)}{\mathbf{H}^i - \mathbf{H}^j}\frac{\ln(\boldsymbol{\Sigma}^i) - \ln(\boldsymbol{\Sigma}^j)}{\boldsymbol{\Sigma}^i - \boldsymbol{\Sigma}^j}) \tag{A.49}$$

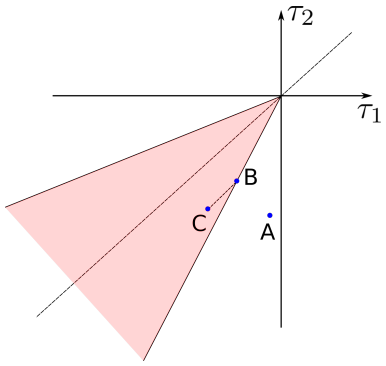


Figure A.1

A.4 Pressure Dependent Yielding: Drucker-Prager

The yielding stresses are pressure dependent for the Drucker-Prager palsticity model. For $\boldsymbol{\Sigma}$ with different sum($\boldsymbol{\Sigma}$) outside the yield surface, the yield stress is different. If we still use (Equation (A.10)) as energy density function, the gradient is then

$$\frac{\partial \Psi}{\partial \boldsymbol{\Sigma}} = (\frac{d\boldsymbol{Z}}{d\boldsymbol{\Sigma}})^\top \frac{\partial \Psi^E}{\partial \boldsymbol{\Sigma}^E}(\boldsymbol{Z}) + \tau_Y \frac{d\delta\gamma}{d\boldsymbol{\Sigma}} + \delta\gamma \frac{d\tau}{d\boldsymbol{\Sigma}} \tag{A.50}$$

The relation between the following three vectors

$$\begin{aligned}
\boldsymbol{A} =& \frac{\partial \Psi^E}{\partial \boldsymbol{\Sigma}^E}(\boldsymbol{\Sigma}) \odot \boldsymbol{\Sigma}, \\
\boldsymbol{B} =& [(\frac{d\boldsymbol{Z}}{d\boldsymbol{\Sigma}})^\top \frac{\partial \Psi^E}{\partial \boldsymbol{\Sigma}^E}(\boldsymbol{Z}) + \tau_Y \frac{d\delta\gamma}{d\boldsymbol{\Sigma}}] \odot \boldsymbol{\Sigma}, \\
\mathbf{C} =& [(\frac{d\boldsymbol{Z}}{d\boldsymbol{\Sigma}})^\top \frac{\partial \Psi^E}{\partial \boldsymbol{\Sigma}^E}(\boldsymbol{Z}) + \tau_Y \frac{d\delta\gamma}{d\boldsymbol{\Sigma}} + \delta\gamma \frac{d\tau_Y}{d\boldsymbol{\Sigma}}] \odot \boldsymbol{\Sigma},
\end{aligned} \tag{A.51}$$

can be visualized in Fig Figure A.1.

It can be verified that

$$\forall \boldsymbol{F}, \frac{\partial \Psi^E}{\partial \boldsymbol{F}^E}(\boldsymbol{Z}(\boldsymbol{F}))\boldsymbol{Z}(\boldsymbol{F})^\top \equiv [(\frac{dZ}{d\boldsymbol{F}})^\top \frac{\partial \Psi^E}{\partial \boldsymbol{F}^E}(\boldsymbol{Z}) + \tau_Y \frac{d\delta\gamma}{d\boldsymbol{F}}]\boldsymbol{F}^T. \tag{A.52}$$

According to the proof of Theorem Theorem A.3.2, the MPM force computed from $(\frac{d\boldsymbol{Z}}{d\boldsymbol{F}})^\top \frac{\partial \Psi^E}{\partial \boldsymbol{F}^E}(\boldsymbol{Z}) + \tau_Y(\frac{d\delta\gamma}{d\boldsymbol{F}})$ still coincides with the force from implicit plasticity. However, the Jacobian matrix

of $\mathbf{B}(\mathbf{\Sigma}) = (\frac{d\boldsymbol{Z}}{d\mathbf{\Sigma}})^\top \frac{\partial \Psi^E}{\partial \mathbf{\Sigma}^E}(Z) + \tau_Y \frac{d\delta\gamma}{d\mathbf{\Sigma}}$ is

$$\nabla \mathbf{B} = \frac{d^2\Psi^E(\boldsymbol{Z}(\mathbf{\Sigma}))}{d\mathbf{\Sigma}^2} + \tau_Y \frac{d^2\delta\gamma}{d\mathbf{\Sigma}^2} + (\frac{d\delta\gamma}{d\mathbf{\Sigma}})^\top \frac{d\tau_Y}{d\mathbf{\Sigma}}, \tag{A.53}$$

of which the last term is not symmetric. So $\mathbf{B}$ cannot be the gradient of a smooth energy.

It can be also verified that $\delta\gamma \frac{d\tau_Y}{d\mathbf{\Sigma}} \odot \mathbf{\Sigma}$ is parallel to the central axis of the cone and pointing inwards. So the simulation using MPM force computed from $\frac{\partial \Psi}{\partial \boldsymbol{F}}$ will look like the friction angle is moving towards zero a little, which will produce unrealistic simulation results.

A.4.1 Stress Iteration

We propose to use iterative yielding stresses: run multiple Newton optimizations where before each optimization the yielding stresses are computed and fixed during the optimization.

A.4.1.1 Extrapolated St. Venant-Kirchhoff Model

To define $\delta\gamma$ and yield stress in the area $\text{sum}(\boldsymbol{\epsilon}) > 0$. We introduce the extrapolated St. Venant-Kirchhoff cosntitutive model:

$$\hat{\Psi}^E(\mathbf{\Sigma}) = \begin{cases} \mu\|\hat{\boldsymbol{\epsilon}}\|^2 & \text{tr}(\boldsymbol{\epsilon}) \geq 0 \\ \mu\|\hat{\boldsymbol{\epsilon}}\|^2 + (\frac{\lambda}{2} + \frac{\mu}{d})(\text{tr}(\boldsymbol{\epsilon}))^2 & \text{tr}(\boldsymbol{\epsilon}) < 0 \end{cases}. \tag{A.54}$$

where $\hat{\boldsymbol{\epsilon}}$ is the deviatoric component of the Hencky strain $\boldsymbol{\epsilon}$. $\delta\gamma$ is computed from the volume-preserving projection.

The yielding stress is computed as

$$\tau_Y(\mathbf{\Sigma}) = \begin{cases} 0 & \text{sum}(\boldsymbol{\epsilon}) \geq 0 \\ -\alpha\ \text{sum}(\boldsymbol{\epsilon})(d\lambda + 2\mu) & \text{sum}(\boldsymbol{\epsilon}) < 0 \end{cases}. \tag{A.55}$$

A.4.1.2 Fixed Point Iteration

The goal of iterative stress is to reach the following condition:

$$\boldsymbol{\tau}_Y^* = \boldsymbol{\Gamma}_Y^{tr}(\boldsymbol{F}^{E,tr}(\Delta\mathbf{v}(\boldsymbol{\tau}_Y^*))) \tag{A.56}$$

where τ_Y^*, $\boldsymbol{F}^{E,tr}$ are stacked vectors for particles and $\Delta\mathbf{v}$ is the stacked vector of velocity increments on the grid.

We solve $\boldsymbol{\tau}_Y^*$ by fixed point iteration:

$$\boldsymbol{\tau}_Y^{tr,j+1} = \boldsymbol{\Gamma}_Y^{tr}(\boldsymbol{F}^{E,tr}(\Delta\mathbf{v}(\boldsymbol{\tau}_Y^{tr,j}))) \tag{A.57}$$

To make the iteration converge, one sufficient condition is that $\rho(J_{\boldsymbol{\Gamma}_Y^{tr}}(\boldsymbol{\tau}_Y^{tr,j})) \leq \gamma < 1$, for all j and some constant γ.

At $\boldsymbol{\tau}_Y^{tr,j}$,

$$\mathbf{M}\Delta\mathbf{v} + \Delta t\frac{\partial\Psi}{\partial\Delta\mathbf{v}} = 0 \tag{A.58}$$

Take the derivative w.r.t $\boldsymbol{\tau}_Y$

$$\mathbb{H}_{\Delta\mathbf{v}}\frac{d\Delta\mathbf{v}}{d\boldsymbol{\tau}_Y} + \Delta t^2\frac{\partial^2\Psi}{\partial\Delta\mathbf{v}\partial\boldsymbol{\tau}_Y} = 0, \tag{A.59}$$

that is,

$$\frac{d\Delta\mathbf{v}}{d\boldsymbol{\tau}_Y} = -\Delta t^2\mathbb{H}_{\Delta\mathbf{v}}^{-1}\frac{\partial^2\Psi}{\partial\Delta\mathbf{v}\partial\boldsymbol{\tau}_Y} \tag{A.60}$$

Hence

$$\frac{d\boldsymbol{\Gamma}_Y^{tr}}{d\boldsymbol{\tau}_Y} = -\Delta t^2\frac{\partial\boldsymbol{\Gamma}_Y^{tr}}{\partial\boldsymbol{F}^{E,tr}}\frac{\partial\boldsymbol{F}^{E,tr}}{\partial\Delta\boldsymbol{v}}\mathbb{H}_{\Delta\mathbf{v}}^{-1}\frac{\partial^2\Psi}{\partial\Delta\mathbf{v}\partial\boldsymbol{\tau}_Y} \tag{A.61}$$

As $\Delta t \to 0$, $\mathbb{H}_{\Delta\mathbf{v}} \to \mathbf{M}$, $\frac{\partial\boldsymbol{F}^{E,tr}}{\partial\Delta\boldsymbol{v}} \to 0$, $\rho(J_\tau(\tau^{(n)})) \to 0$.

A.4.2 Implementation Tricks

When $\text{sum}(\boldsymbol{\epsilon}) < 0$, we compute $\frac{\partial\Psi}{\partial\boldsymbol{\Sigma}}$ and $\frac{\partial^2\Psi}{\partial\boldsymbol{\Sigma}^2}$ as we do for von-Mises plasticity.

When $\text{sum}(\boldsymbol{\epsilon}) \geq 0$,

$$\frac{\partial \Psi}{\partial \boldsymbol{\Sigma}} = 2\mu(\mathbf{H} - \frac{1}{d}\,\text{sum}(\mathbf{H})\mathbf{1}) \odot \boldsymbol{\Sigma}^{-1} \tag{A.62}$$

$$\frac{\partial^2 \Psi}{\partial \boldsymbol{\Sigma}^2} = 2\mu\,\text{diag}(\boldsymbol{\Sigma}^{-1})(\boldsymbol{I} - \frac{1}{d}\mathbf{1}\mathbf{1}^\top)\frac{d\mathbf{H}}{d\boldsymbol{\Sigma}} - 2\mu\text{diag}(\mathbf{H} - \frac{1}{d}\,\text{sum}(\mathbf{H})\mathbf{1})\ \text{diag}(\boldsymbol{\Sigma}^{-2}) \tag{A.63}$$

The coefficients needed to evaluate hessian w.r.t. deformation gradient in this region are

$$\frac{\partial_{\boldsymbol{\Sigma}^i}\Psi - \partial_{\boldsymbol{\Sigma}^j}\Psi}{\boldsymbol{\Sigma}^i - \boldsymbol{\Sigma}^j} = \frac{2\mu}{\boldsymbol{\Sigma}^i\boldsymbol{\Sigma}^j}(-\mathbf{H}^i + \boldsymbol{\Sigma}^i\frac{\ln(\boldsymbol{Z}^i) - \ln(\boldsymbol{Z}^j)}{\boldsymbol{Z}^i - \boldsymbol{Z}^j}\frac{\boldsymbol{Z}^i - \boldsymbol{Z}^j}{\boldsymbol{\Sigma}^i - \boldsymbol{\Sigma}^j} + \frac{1}{d}\,\text{sum}(\mathbf{H})) \tag{A.64}$$

A.5 Viscoelasticity

For viscoelasticity, the return mapping is

$$\mathbf{H} = A(\boldsymbol{\epsilon} - B\,\text{sum}(\boldsymbol{\epsilon})\mathbf{1}) \tag{A.65}$$

According to Lemma Lemma A.3.1, we can assume

$$\frac{\partial \Psi}{\partial \boldsymbol{\Sigma}} = \frac{\partial \Psi^E}{\partial \boldsymbol{\Sigma}^E}(\mathbf{Z}(\boldsymbol{\Sigma})) \odot \mathbf{Z}(\boldsymbol{\Sigma}) \odot \boldsymbol{\Sigma}^{-1} \tag{A.66}$$

Plug in the definition of Ψ^E:

$$\begin{aligned}\frac{\partial \Psi}{\partial \boldsymbol{\Sigma}} &= (2\mu\mathbf{H} + \lambda\,\text{sum}(\mathbf{H})\mathbf{1}) \odot \boldsymbol{\Sigma}^{-1} \\ &= (2\mu(A(\boldsymbol{\epsilon} - B\,\text{sum}(\boldsymbol{\epsilon})\mathbf{1})) + \lambda A(1 - dB)\,\text{sum}(\boldsymbol{\epsilon})\mathbf{1}) \odot \boldsymbol{\Sigma}^{-1} \\ &= (2A\mu\boldsymbol{\epsilon} + (A\lambda - AB(2\mu + d\lambda))\,\text{sum}(\boldsymbol{\epsilon})\mathbf{1}) \odot \boldsymbol{\Sigma}^{-1}\end{aligned} \tag{A.67}$$

Then

$$\frac{\partial \Psi}{\partial \boldsymbol{\Sigma}} = (2\hat{\mu}\boldsymbol{\epsilon} + \hat{\lambda}\,\text{sum}(\boldsymbol{\epsilon})\mathbf{1}) \odot \boldsymbol{\Sigma}^{-1} \tag{A.68}$$

where $\hat{\mu} = A\mu$, $\hat{\lambda} = A\lambda - AB(2\mu + d\lambda)$.

This vector field can be integrated to get $\Psi(\boldsymbol{\Sigma}) = \Psi^E(\boldsymbol{\Sigma}\big|\hat{\mu}, \hat{\lambda})$.

A.6 Discretization with FEM

For FEM with linear tetrahedral elements, the discretized nodal internal force is

$$\boldsymbol{f}_i = -\sum_e \boldsymbol{P}_e \nabla N_{ie} V_e^0, \tag{A.69}$$

where e indices all tetrahedral elements, V_e is the rest volume of element e, and ∇N_{ie} is the gradient of the shape function on node i evaluated at the barycenter of element e.

By plugging in $\boldsymbol{P} = \frac{\partial \Psi^E}{\partial \boldsymbol{F}^E} \boldsymbol{F}^{P^{-\top}}$, the nodal force with implicit plasticity is

$$\boldsymbol{f}_i^{n+1} = -\sum_e V_e^0 \frac{\partial \Psi^E}{\partial \boldsymbol{F}^E}(F_e^{E,n+1}) \boldsymbol{F}_e^{P,n+1^{-\top}} \nabla N_{ie}. \tag{A.70}$$

Note that,

$$\boldsymbol{F}^{E,n+1} = \mathbf{Z}(\boldsymbol{F}^{E,tr}), \tag{A.71}$$

$$\boldsymbol{F}^{P,n+1} = \boldsymbol{F}^{E,n+1^{-1}} \boldsymbol{F}^{n+1} = \boldsymbol{F}^{E,n+1^{-1}} \boldsymbol{F}^{E,tr} \boldsymbol{F}^{P,n}. \tag{A.72}$$

The internal force with implicit plasticity can be written as

$$\begin{aligned} \boldsymbol{f}_i^{n+1} &= -\sum_e V_e^0 \frac{\partial \Psi^E}{\partial \boldsymbol{F}^E}(\boldsymbol{F}_e^{E,n+1}) \boldsymbol{F}_e^{E,n+1^T} \boldsymbol{F}_e^{E,tr^{-\top}} \boldsymbol{F}^{P,n^{-\top}} \nabla N_{ie}, \\ &= -\sum_e V_e^0 \frac{\partial \Psi^E}{\partial \boldsymbol{F}^E}(\mathbf{Z}(\boldsymbol{F}_e^{E,tr})) \mathbf{Z}(\boldsymbol{F}_e^{E,tr})^T \boldsymbol{F}_e^{E,tr^{-\top}} \boldsymbol{F}^{P,n^{-\top}} \nabla N_{ie} \end{aligned} \tag{A.73}$$

In tetrahedral FEM,

$$\boldsymbol{F} = \boldsymbol{T}\boldsymbol{B}^{-1} \tag{A.74}$$

where $\boldsymbol{T} = [\boldsymbol{x}_1 - \boldsymbol{x}_0, \boldsymbol{x}_2 - \boldsymbol{x}_0, \boldsymbol{x}_3 - \boldsymbol{x}_0]$ and $\boldsymbol{B} = [\boldsymbol{X}_1 - \boldsymbol{X}_0, \boldsymbol{X}_2 - \boldsymbol{X}_0, \boldsymbol{X}_3 - \boldsymbol{X}_0]$ are the triangle bases of the element in the world space and the material space respectively.

So the elastic predictor is computed as

$$\boldsymbol{F}^{E,tr} = \boldsymbol{F}\boldsymbol{F}^{P,n^{-1}} = \boldsymbol{T}\boldsymbol{B}^{-1}\boldsymbol{F}^{P,n^{-1}}, \tag{A.75}$$

The integrability of the vector field $\frac{\partial \Psi^E}{\partial \boldsymbol{F}^E}(\mathcal{Z}(\boldsymbol{F}^{E,tr}))\mathcal{Z}(\boldsymbol{F}^{E,tr})^\top \boldsymbol{F}^{E,tr^{-\top}}$ leads us to the integrable internal force from the augmented elastoplastic energy density Ψ:

$$\boldsymbol{f}_i^{n+1} = -\sum_e V_e^0 \frac{\partial \Psi}{\partial \boldsymbol{F}^{E,tr}} \frac{\partial \boldsymbol{F}^{E,tr}}{\partial \boldsymbol{x}_i} = -\frac{\partial}{\partial \boldsymbol{x}_i}(\sum_e V_e^0 \Psi(\boldsymbol{F}^{E,tr})), \tag{A.76}$$

where x_i is the world space coordinate of node i.

In FEM, we need to track $\boldsymbol{F}^P$ on elements, because $\boldsymbol{F}^E$ is not stored across time steps. $\boldsymbol{F}^P$ is updated by the relation:

$$\mathbf{Z}(\boldsymbol{F}^{E,tr})\boldsymbol{F}^{P,n+1} = \boldsymbol{F}^{E,tr}\boldsymbol{F}^{P,n} \tag{A.77}$$

at the end of each time step.

For the simplicity of implementation, we can directly update the triangle bases of the rest shape. For readability, we omit the element index e in the following derivations.

Define

$$\boldsymbol{B}^n = \boldsymbol{F}^{P,n}\boldsymbol{B}. \tag{A.78}$$

The update rule (Equation (A.77)) now becomes:

$$\mathbf{Z}(\boldsymbol{F}^{E,tr})\boldsymbol{B}^{n+1} = \boldsymbol{F}^{E,tr}\boldsymbol{B}^n. \tag{A.79}$$

For isotropic materials, let

$$\boldsymbol{F}^{E,tr} = \boldsymbol{U}\boldsymbol{\Sigma}^{E,tr}\boldsymbol{V}^\top,$$

$$\mathbf{Z}(\boldsymbol{F}^{E,tr}) = \boldsymbol{U}\hat{\mathbf{Z}}\boldsymbol{V}^\top.$$

The update rule for $\boldsymbol{B}$ is then:

$$\boldsymbol{B}^{n+1} = \boldsymbol{V}\hat{\mathbf{Z}}^{-1}\boldsymbol{\Sigma}^{E,tr}\boldsymbol{V}^\top\boldsymbol{B}^n \tag{A.80}$$

APPENDIX B

Elasticity Models and Plasticity Models

Table B.1: Material parameter notations.

Notation	Meaning	Relation to E, ν
E	Young's modulus	/
ν	Poisson's ratio	/
μ	Shear modulus	$\mu = \frac{E}{2(1+\nu)}$
λ	Lamé modulus	$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}$
κ	Bulk modulus	$\kappa = \frac{E}{3(1-2\nu)}$

Here we list elasticity models and plasticity models used in this thesis. The material parameters are defined in Table B.1.

B.1 Elasticity Models

B.1.1 Fixed Corotated Elasticity

The energy density function is defined as

$$\Psi(\boldsymbol{F}) = \mu\|F - R\|_F^2 + \frac{\lambda}{2}(J-1)^2. \tag{B.1}$$

The first-Piola Kirchhoff stress $\boldsymbol{P} = \frac{\partial \Psi}{\partial \boldsymbol{F}}$ is

$$\boldsymbol{P} = 2\mu(\boldsymbol{F} - \boldsymbol{R} + \lambda(J-1)JF^{-T}, \tag{B.2}$$

where $\boldsymbol{R} = \boldsymbol{U}\boldsymbol{V}^T$ and $\boldsymbol{F} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ is the singular value decomposition of elastic deformation gradient. J is the determinant of $\boldsymbol{F}$.

B.1.2 Neo-Hookean Elasticity

The energy density function is defined as

$$\Psi(\boldsymbol{F}) = \frac{\mu}{2}(\mathrm{tr}(\boldsymbol{F}^T\boldsymbol{F}) - d) - \mu\log(J) + \frac{\lambda}{2}\log(J)^2. \tag{B.3}$$

The first-Piola Kirchhoff stress $\boldsymbol{P}$ is defined as

$$\boldsymbol{P} = \mu(\boldsymbol{F} - \boldsymbol{F}^T) + \lambda\log(J)\boldsymbol{F}^{-T}, \tag{B.4}$$

B.1.3 StVK Elasticity

The energy density function is defined as

$$\Psi(\boldsymbol{F}) = \mu\,\mathrm{tr}((\log\boldsymbol{\Sigma}))^2 + \frac{\lambda}{2}(\mathrm{tr}(\log\boldsymbol{\Sigma}))^2. \tag{B.5}$$

The Kirchhoff stress $\boldsymbol{\tau}$ is defined as

$$\boldsymbol{P} = \boldsymbol{U}\left(2\mu\boldsymbol{\epsilon} + \lambda\,\mathrm{sum}(\boldsymbol{\epsilon})\mathbf{1}\right)\boldsymbol{\Sigma}^{-1}\boldsymbol{V}^T, \tag{B.6}$$

where $\boldsymbol{\epsilon} = \log(\boldsymbol{\Sigma})$ and $\boldsymbol{F}^E = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$.

Newtonian Fluid MPM can simulate compressible fluids as well. We use J-based fluid combined with viscosity term to model Newtonian fluid. The Kirchhoff stress $\boldsymbol{\tau}$ for this model is defined by

$$\boldsymbol{\tau} = \frac{1}{2}\mu(\nabla\boldsymbol{v} + \nabla\boldsymbol{v}^\top) + \kappa(J - \frac{1}{J^6})\boldsymbol{I}. \tag{B.7}$$

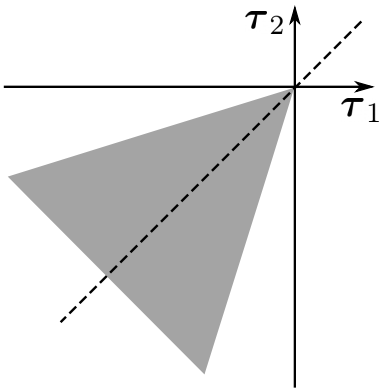


Figure B.1: Drucker-Prager plasticity's elastic region in the stress space.

B.2 Plasticity Models

For isotropic materials, we can define return mappings on the singular values of $\boldsymbol{F}$. The full return mapping is then obtained by reconstructing the matrix from the singular value space:

$$\mathcal{Z}(\boldsymbol{F}) = \boldsymbol{U}\,\mathrm{Diag}(\mathcal{Z}(\boldsymbol{\Sigma}))\boldsymbol{V} \tag{B.8}$$

B.2.1 Sand Plasticity

We use StVK elasticity and Drucker-Prager plasticity for sand simulations (Klár et al., 2016).

The elastic region is characterised in the stress space as:

$$\alpha\,\mathrm{sum}(\tau) + \|\,\mathrm{sum}(\tau) - \frac{\mathrm{sum}(\tau)}{d}\mathbf{1}\| \leq 0, \tag{B.9}$$

where $\alpha = \sqrt{\frac{2}{3}}\frac{2\sin\phi_f}{3-\sin\phi_f}$ and ϕ_f is the friction angle. In our sand examples, ϕ_f is set to $\frac{\pi}{6}$. The elastic region in the stress space is shown in Figure B.1.

The return mapping for the Drucker-Prager plasticity is

$$\mathcal{Z}(\boldsymbol{\Sigma}) = \begin{cases} \mathbf{1} & \mathrm{sum}(\boldsymbol{\epsilon}) > 0 \\ \boldsymbol{\Sigma} & \delta\gamma \leq 0, \text{and}\,\mathrm{sum}(\boldsymbol{\epsilon}) \leq 0\,, \\ \exp\left(\boldsymbol{\epsilon} - \delta\gamma\frac{\hat{\boldsymbol{\epsilon}}}{\|\hat{\boldsymbol{\epsilon}}\|}\right) & \text{otherwise} \end{cases} \tag{B.10}$$

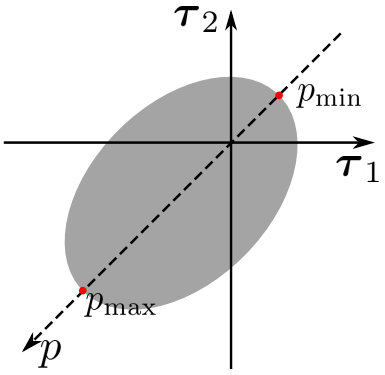


Figure B.2: NACC plasticity's elastic region in the stress space.

where $\delta\gamma = \|\hat{\boldsymbol{\epsilon}}\| + \alpha\frac{(d\lambda+2\mu)\,\mathrm{sum}(\boldsymbol{\epsilon})}{2\mu}$.

B.2.2 Snow Plasticity

We use a specific form of neo-Hookean elasticity and non-associative Cam-Clay (NACC) plasticity for snow simulations (Wolper et al., 2019).

The Kirchhoff stress of neo-Hookean elasticity is

$$\begin{aligned} J &= \det \mathrm{Diag}(\boldsymbol{\Sigma}), \\ \mathbf{b} &= \boldsymbol{\Sigma}^2, \\ \widehat{\mathbf{b}} &= \mathrm{dev}(\mathbf{b}) = \mathbf{b} - \frac{\mathrm{sum}(\mathbf{b})}{d}\mathbf{1}, \\ \boldsymbol{\tau} &= \mu J^{-\frac{2}{d}}\widehat{\mathbf{b}} + \frac{K}{2}(J^2-1)\mathbf{1}. \end{aligned} \tag{B.11}$$

The elastic region of NACC is characterized by

$$y(p,q) = q^2(1+2\beta) + M^2(p+\beta p_0)(p-p_0) \le 0, \tag{B.12}$$

where

$$
\begin{aligned}
M &= d\sqrt{\frac{6-d}{2}}\sqrt{\frac{2}{3}}\frac{2\sin\phi_f}{3-\sin\phi_f} \\
p &= \frac{K}{2}(J^2-1), \\
q &= \mu J^{-\frac{2}{d}}\sqrt{\frac{6-d}{2}}\|\widehat{\mathbf{b}}\|, \\
p_0 &= K\sinh(\xi\max\{-\alpha,0\}).
\end{aligned}
\tag{B.13}
$$

ξ, β, ϕ_f are the parameters of plasticity and α is the hardening state. The elastic region in the stress space is shown in Figure B.2. In our snow examples, $\xi = 0.5$, $\beta = 0.3$ and $\phi_f = \frac{\pi}{4}$.

The return mapping is defined as

$$
\mathcal{Z}(\mathbf{\Sigma}) = \begin{cases}
(1-\frac{2p_{\max}}{K})^{-\frac{1}{2d}}\mathbf{1}, & p > p_{\max} = p_0, \\
(1+\frac{2p_{\min}}{K})^{-\frac{1}{2d}}\mathbf{1}, & p < p_{\min} = -\beta p_0, \\
\mathbf{\Sigma}, & y(p,q) \le 0, \\
\frac{J^{-\frac{2}{d}}}{\mu}\sqrt{\frac{-2M^2(p+\beta p_0)(p-p_0)}{(6-d)(1+2\beta)}}\frac{\widehat{\mathbf{b}}}{\|\widehat{\mathbf{b}}\|} + \frac{1}{d}\operatorname{sum}(\mathbf{b})\mathbf{1} & \text{Otherwise}
\end{cases}
\tag{B.14}
$$

Please refer to (Wolper et al., 2019) for the hardening state update procedure, which is controlled by the simulator. For PlasticityNet, we set $h = \min\{\alpha, 0\}$ as the hardening state input.

B.2.3 Metal Plasticity under StVK Elasticity

We use StVK elasticity and von-Mises plasticity for metal simulations (Klár et al., 2016). This combination provides a closed-form return mapping projection. This plastic model without hardening is also used to simulate plasticine in PAC-NeRF (Section 6.1).

The elastic region is characterized by

$$
\|\boldsymbol{\tau} - \frac{1}{d}\operatorname{sum}(\boldsymbol{\tau})\| - \tau_Y \le 0,
\tag{B.15}
$$

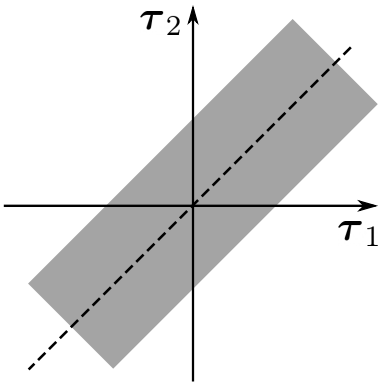


Figure B.3: Von-Mises plasticity's elastic region in the stress space.

where τ_Y controls the radius of the yield surface in the stress space (Figure B.3).

The return mapping for the von-Mises plasticity is defined as

$$\mathcal{Z}(\boldsymbol{\Sigma}) = \begin{cases} \boldsymbol{\Sigma}, & \|\boldsymbol{\tau} - \frac{1}{d}\operatorname{sum}(\boldsymbol{\tau})\| - \tau_Y \leq 0 \\ \exp\left(\boldsymbol{\epsilon} - \delta\gamma \frac{\hat{\boldsymbol{\epsilon}}}{\|\hat{\boldsymbol{\epsilon}}\|}\right), & \text{Otherwise} \end{cases}, \tag{B.16}$$

where $\delta\gamma = \|\hat{\boldsymbol{\epsilon}}\|_F - \frac{\tau_Y}{2\mu}$.

Under hardening, τ_Y is updated with

$$\tau_Y^{n+1} = \tau_Y^n + 2\mu\xi\delta\gamma, \tag{B.17}$$

where ξ is the hardening coefficient.

Non-Newtonian Fluid Non-Newtonian fluid also has a yield stress. We use viscoplastic model (Yue et al., 2015) to model non-Newtonian fluid. We still use von-Mises criteria to define elastic region. However, the existence of viscoplastic means the deformation will not be directly projected back onto the yield surface. Define

$$\begin{aligned} \hat{\mu} &= \frac{\mu}{d}\operatorname{Tr}(\boldsymbol{\Sigma}^2), \\ \boldsymbol{s} &= 2\mu\hat{\epsilon}, \\ \hat{s} &= \|\boldsymbol{s}\| - \frac{\delta\gamma}{1 + \frac{\eta}{2\hat{\mu}\Delta t}}, \end{aligned} \tag{B.18}$$

The return mapping is defined as

$$\mathcal{Z}(\boldsymbol{F}) = \begin{cases} \boldsymbol{F}, & \delta\gamma \leq 0, \\ \boldsymbol{U} \exp\left(\frac{\hat{s}}{2\mu}\hat{\boldsymbol{\epsilon}} + \frac{1}{d}\operatorname{Tr}(\boldsymbol{\epsilon})\mathbf{1}\right)\boldsymbol{V}^\top, & \text{otherwise.} \end{cases} \tag{B.19}$$

B.2.4 Metal Plasticity under Neo-Hookean Elasticity

The combination of neo-Hookean elasticity and von-Mises plasticity does not have a closed-form return mapping. In PlasitictyNet (Section 3.2), we use this combination for the task of learning metal plasticity return mapping. The Kirchhoff stress of neo-Hookean elasticity is given by

$$\boldsymbol{\tau} = \mu(\boldsymbol{\Sigma}^2 - \mathbf{1}) + \lambda \log J\mathbf{1}. \tag{B.20}$$

The implicit representation of the elastic region we used in the training of the return mapping is given by

$$y(\boldsymbol{\Sigma}, \tau_Y) = \|\boldsymbol{\tau} - \frac{1}{d}\operatorname{sum}(\boldsymbol{\tau})\|^2 - \tau_Y^2. \tag{B.21}$$

Aage, N., Andreassen, E., Lazarov, B. S., and Sigmund, O. (2017). Giga-voxel computational morphogenesis for structural design. *Nature*, 550(7674):84. 282, 290

Abe, K., Soga, K., and Bandara, S. (2014). Material point method for coupled hydromechanical problems. *Journal of Geotechnical and Geoenvironmental Engineering*, 140(3):04013033. 73, 285

Abu Rumman, N., Nair, P., Müller, P., Barthe, L., and Vanderhaeghe, D. (2020). Isph–pbd: coupled simulation of incompressible fluids and deformable bodies. *The Visual Computer*, 36(5):893–910. 162

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. (2023). Gpt-4 technical report. *arXiv:2303.08774*. 270

Alduán, I. and Otaduy, M. A. (2011). Sph granular flow with friction and cohesion. In *Proceedings of the 2011 Symp. Computer animation*, pages 25–32. 19, 164

Allaire, G., Jouve, F., and Toader, A.-M. (2004). Structural optimization using sensitivity analysis and a level-set method. *Journal of computational physics*, 194(1):363–393. 282

An, S. S., Kim, T., and James, D. L. (2008). Optimizing cubature for efficient integration of subspace deformations. *ACM transactions on graphics (TOG)*, 27(5):1–10. 140

Ando, R., Thurey, N., and Tsuruno, R. (2012). Preserving fluid sheets with adaptively sampled anisotropic particles. *IEEE Transactions on Visualization and Computer Graphics*, 18(8):1202–1214. 173

Ando, R. and Tsuruno, R. (2011). A particle-based method for preserving fluid sheets. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics symposium on computer animation*, pages 7–16. 173

Andreassen, E., Clausen, A., Schevenels, M., Lazarov, B. S., and Sigmund, O. (2011). Efficient topology optimization in matlab using 88 lines of code. *Structural and Multidisciplinary Optimization*, 43(1):1–16. 282, 287, 321

Andrews, S., Erleben, K., and Ferguson, Z. (2022). Contact and friction simulation for computer graphics. In *ACM SIGGRAPH 2022 Courses*, pages 1–172. 139

Antić, D., Tiwari, G., Ozcomlekci, B., Marin, R., and Pons-Moll, G. (2024). Close: A 3d clothing segmentation dataset and model. In *2024 International Conference on 3D Vision (3DV)*, pages 591–601. IEEE. 271

ARRISHobby. Arris x220 v2 220mm 5" fpv racing quad rtf w/radiolink at9s. https://arrishobby.com/arris-x220-v2-220mm-5inch-fpv-racing-quad-rtf-p0017.html. Accessed: 2020-08-06. 330

ARRISHobby. Arris x2206 1500kv 2450kv 3-4s brushless motor for fpv racing drones. https://www.arrishobby.com/arris-x2206-2450kv-3-4s-brushless-motor-for-fpv-racing-drones-p0193.html. Accessed: 2020-08-06. 330

As' ad, F., Avery, P., and Farhat, C. (2022). A mechanics-informed artificial neural network approach in data-driven constitutive modeling. In *AIAA Scitech 2022 Forum*, page 0100. 56

Aulisa, E. and Capodaglio, G. (2019). Monolithic coupling of the implicit material point method with the finite element method. *Computers & Structures*, 219:1–15. 75

Bandara, S. and Soga, K. (2015). Coupling of soil deformation and pore fluid flow using material point method. *Computers and geotechnics*, 63:199–214. 285

Banerjee, B. (2005). Simulation of thin hyperelastic shells with the material point method. 74

Bangor, A., Kortum, P., and Miller, J. (2009). Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, 4(3):114–123. 244, 246

Bao, C., Zhang, Y., Yang, B., Fan, T., Yang, Z., Bao, H., Zhang, G., and Cui, Z. (2023). Sine: Semantic-driven image-based nerf editing with prior-guided editing field. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20919–20929. 227

Baraff, D. (1994). Fast contact force computation for nonpenetrating rigid bodies. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 23–34. 139

Baraff, D. and Witkin, A. (1998). Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 43–54. 113, 136, 138, 139

Barbarino, S., Bilgen, O., Ajaj, R. M., Friswell, M. I., and Inman, D. J. (2011). A review of morphing aircraft. *Journal of intelligent material systems and structures*, 22(9):823–877. 318

Barber, J. R. (2009). Plane contact problems. In *Solid Mechanics and Its Applications*, pages 171–198. Springer Netherlands. 100

Barbič, J. and Zhao, Y. (2011). Real-time large-deformation substructuring. *ACM transactions on graphics (TOG)*, 30(4):1–8. 140

Barbič, J. and James, D. L. (2005). Real-time subspace integration for st. venant-kirchhoff deformable models. In *ACM Trans. Graph. (TOG)*, volume 24, pages 982–990. ACM. 140

Bardenhagen, S., Brackbill, J., and Sulsky, D. (2000). The material-point method for granular materials. *Computer methods in applied mechanics and engineering*, 187(3-4):529–541. 285

Bardenhagen, S. G. and Kober, E. M. (2004). The generalized interpolation material point method. *Computer Modeling in Engineering and Sciences*, 5(6):477–496. 73, 85

Bargteil, A. W., C. Wojtan, J. K. H., and Turk, G. (2007). A finite element method for animating large viscoplastic flow. *ACM Trans. on Graph.*, 26(3). 18, 110, 113

Bargteil, A. W., Shinar, T., and Kry, P. G. (2020). An introduction to physics-based animation. In *SIGGRAPH Asia 2020 Courses*, pages 1–57. 54

Barreiro, H., García-Fernández, I., Alduán, I., and Otaduy, M. A. (2017). Conformation constraints for efficient viscoelastic fluid simulation. *ACM Transactions on Graphics (TOG)*, 36(6):1–11. 160, 163, 184

Barron, J. T., Mildenhall, B., Verbin, D., Srinivasan, P. P., and Hedman, P. (2022). Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479. 207

Bartle, A., Sheffer, A., Kim, V. G., Kaufman, D. M., Vining, N., and Berthouzoz, F. (2016). Physics-driven pattern adjustment for direct 3d garment editing. *ACM Trans. Graph.*, 35(4):50–1. 252

Battaglia, P., Pascanu, R., Lai, M., Jimenez Rezende, D., et al. (2016). Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29. 54

Batty, C., Bertails, F., and Bridson, R. (2007). A fast variational framework for accurate solid-fluid coupling. *ACM Trans. on Graph.*, 26(3). 21, 286

Becker, M. and Sheffler, D. (2016). Designing a high speed, stealthy, and payload-focused vtol uav. In *2016 IEEE Systems and Information Engineering Design Symposium (SIEDS)*, pages 176–180. IEEE. 318

Belbute-Peres, F. D. A., Economon, T., and Kolter, Z. (2020). Combining differentiable pde solvers and graph neural networks for fluid flow prediction. In *International Conference on Machine Learning*, pages 2402–2411. PMLR. 54

Bender, J., Koschier, D., Charrier, P., and Weber, D. (2014). Position-based simulation of continuous materials. *Computers & Graphics*, 44:1–10. 159, 162

Bender, J., Müller, M., and Macklin, M. (2017). A survey on position based dynamics, 2017. In *European Association for Computer Graphics: Tutorials*, pages 1–31. 20, 162

Bergou, M., Wardetzky, M., Harmon, D., Zorin, D., and Grinspun, E. (2006). A quadratic bending model for inextensible surfaces. In *Symposium on Geometry Processing*, pages 227–230. 139, 141

Bergou, M., Wardetzky, M., Robinson, S., Audoly, B., and Grinspun, E. (2008). Discrete elastic rods. In *ACM SIGGRAPH 2008 papers*, pages 1–12. 110, 113

Bewick, B. T. (2004). *A combined FEM and MPM simulation of impact-resistant design*. University of Missouri-Columbia. 75

Bock, F. E., Aydin, R. C., Cyron, C. J., Huber, N., Kalidindi, S. R., and Klusemann, B. (2019). A review of the application of machine learning and data mining approaches in continuum materials mechanics. *Frontiers in Materials*, page 110. 54

Bonet, J. and Lok, T.-S. (1999). Variational and momentum preservation aspects of smooth particle hydrodynamic formulations. *Computer Methods in applied mechanics and engineering*, 180(1-2):97–115. 166, 167

Bonet, J. and Wood, R. D. (1997). *Nonlinear continuum mechanics for finite element analysis*. Cambridge university press. 26, 210, 211, 321, 337

Bonet, J. and Wood, R. D. (2008). *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press. 77

Bouaziz, S., Martin, S., Liu, T., Kavan, L., and Pauly, M. (2014). Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. on Graph.*, 33(4). 8, 21, 113, 137, 164, 252

Brackbill, J. U., Kothe, D. B., and Ruppel, H. M. (1988). Flip: a low-dissipation, particle-in-cell method for fluid flow. *Computer Physics Communications*, 48(1):25–38. 73, 85, 283

Brackbill, J. U. and Ruppel, H. M. (1986). Flip: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational Physics*, 65(2). 18, 173

Brandt, C., Eisemann, E., and Hildebrandt, K. (2018). Hyper-reduced projective dynamics. *ACM Transactions on Graphics (TOG)*, 37(4):1–13. 140, 151, 152

Breen, D. E., House, D. H., and Wozny, M. J. (1994). Predicting the drape of woven cloth using interacting particles. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 365–372. 139

Brezillon, P., Staub, J.-F., Perault-Staub, A.-M., and Milhaud, G. (1981). Numerical estimation of the first order derivative: approximate evaluation of an optimal step. *Computers & Mathematics with Applications*, 7(4):333–347. 252

Bridson, R. (2007). Fast poisson disk sampling in arbitrary dimensions. *SIGGRAPH sketches*, 10(1):1. 172

Bridson, R., Fedkiw, R., and Anderson, J. (2002). Robust treatment of collisions, contact and friction for cloth animation. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 594–603. 139

Bridson, R., Marino, S., and Fedkiw, R. (2005). Simulation of clothing with folds and wrinkles. In *ACM SIGGRAPH 2005 Courses*, pages 3–es. 139

Brochu, T., Edwards, E., and Bridson, R. (2012). Efficient geometrically exact continuous collision detection. *ACM Transactions on Graphics (TOG)*, 31(4):1–7. 91

Brown, G., Overby, M., Forootaninia, Z., and Narain, R. (2018). Accurate dissipative forces in optimization integrators. *ACM Trans. on Graph.*, 37(6):1–14. 22

Brown, G. E. and Narain, R. (2021). Wrapd: weighted rotation-aware admm for parameterization and deformation. *ACM Trans. on Graph.*, 40(4):1–14. 21

Bruns, T. and Tortorelli, D. (1998). Topology optimization of geometrically nonlinear structures and compliant mechanisms. American Institute of Aeronautics and Astronautics. 314

Bucki, N. and Mueller, M. W. (2019). Design and control of a passively morphing quadcopter. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9116–9122. IEEE. 318

Buhl, T., Pedersen, C. B., and Sigmund, O. (2000). Stiffness design of geometrically nonlinear structures using topology optimization. *Structural and Multidisciplinary Optimization*, 19(2):93–104. 285, 314

Burghardt, J., Brannon, R., and Guilkey, J. (2012). A nonlocal plasticity formulation for the material point method. *Computer methods in applied mechanics and engineering*, 225:55–64. 285

Capell, S., Green, S., Curless, B., Duchamp, T., and Popović, Z. (2002). Interactive skeleton-driven dynamic deformations. In *ACM Trans. Graph. (TOG)*, volume 21, pages 586–593. ACM. 140

Cen, J., Fang, J., Yang, C., Xie, L., Zhang, X., Shen, W., and Tian, Q. (2023). Segment any 3d gaussians. *arXiv preprint arXiv:2312.00860*. 223

Chakrabarty, J. and Drugan, W. (1988). Theory of plasticity. 36

Chandra, B., Singer, V., Teschemacher, T., Wuechner, R., and Larese, A. (2021). Non-conforming dirichlet boundary conditions in implicit material point method by means of penalty augmentation. *Acta Geotechnica*, pages 1–21. 76

Chandrasekhar, S. (1967). Ellipsoidal figures of equilibrium—an historical account. *Communications on Pure and Applied Mathematics*, 20(2):251–265. 214

Chang, M. B., Ullman, T., Torralba, A., and Tenenbaum, J. B. (2016). A compositional object-based approach to learning physical dynamics. *arXiv preprint arXiv:1612.00341*. 54

Charlton, T., Coombs, W., and Augarde, C. (2017). igimp: An implicit generalised interpolation material point method for large deformations. *Computers & Structures*, 190:108–125. 76, 285

Chen, C.-H., Su, J.-W., Hu, M.-C., Yao, C.-Y., and Chu, H.-K. (2024a). Panelformer: Sewing pattern reconstruction from 2d garment images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 454–463. 251

Chen, H., Shen, B., Liu, Y., Shi, R., Zhou, L., Lin, C. Z., Gu, J., Su, H., Wetzstein, G., and Guibas, L. (2024b). 3d-adapter: Geometry-consistent multi-view diffusion for high-quality 3d generation. *arXiv preprint arXiv:2410.18974*. 249

Chen, H., Zhang, Y., Cun, X., Xia, M., Wang, X., Weng, C., and Shan, Y. (2024c). Videocrafter2: Overcoming data limitations for high-quality video diffusion models. In *Computer Vision and Pattern Recognition (CVPR)*, pages 7310–7320. 249

Chen, H.-y., Tretschk, E., Stuyck, T., Kadlecek, P., Kavan, L., Vouga, E., and Lassner, C. (2022a). Virtual elastic objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15827–15837. 190, 195, 200, 201, 208

Chen, J., Zhang, Y., Kang, D., Zhe, X., Bao, L., Jia, X., and Lu, H. (2021a). Animatable neural radiance fields from monocular rgb videos. *arXiv preprint arXiv:2106.13629*. 227

Chen, J.-K., Lyu, J., and Wang, Y.-X. (2023a). Neuraleditor: Editing neural radiance fields via manipulating point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12439–12448. 208, 214

Chen, P. Y., Chantharayukhonthorn, M., Yue, Y., Grinspun, E., and Kamrin, K. (2021b). Hybrid discrete-continuum modeling of shear localization in granular media. *Journal of the Mechanics and Physics of Solids*, page 104404. 74

Chen, Q., Zhang, X., Zhang, H., Zhu, B., and Chen, B. (2019). Topology optimization of bistable mechanisms with maximized differences between switching forces in forward and backward direction. *Mechanism and Machine Theory*, 139:131–143. 320, 322

Chen, Q., Zhang, X., and Zhu, B. (2018a). Design of buckling-induced mechanical metamaterials for energy absorption using topology optimization. *Structural and Multidisciplinary Optimization*, 58(4):1395–1410. 320, 322

Chen, W., Zhu, F., Zhao, J., Li, S., and Wang, G. (2018b). Peridynamics-based fracture animation for elastoplastic solids. In *Computer Graphics Forum*, volume 37. 20, 164

Chen, Y., Chen, Z., Zhang, C., Wang, F., Yang, X., Wang, Y., Cai, Z., Yang, L., Liu, H., and Lin, G. (2023b). Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. *arXiv preprint arXiv:2311.14521*. 227

Chen, Y., Davis, T. A., Hager, W. W., and Rajamanickam, S. (2008). Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software (TOMS)*, 35(3):1–14. 93, 94

Chen, Y., Han, Y., Chen, J., Ma, S., Fedkiw, R., and Teran, J. (2023c). Primal extended position based dynamics for hyperelasticity. In *Proceedings of the 16th ACM SIGGRAPH Conference on Motion, Interaction and Games*, pages 1–10. 159

Chen, Y., Li, M., Lan, L., Su, H., Yang, Y., and Jiang, C. (2022b). A unified newton barrier method for multibody dynamics. *ACM Trans. Graph. (SIGGRAPH)*, 41(4). 57

Chen, Y., Li, M., Lan, L., Su, H., Yang, Y., and Jiang, C. (2022c). A unified newton barrier method for multibody dynamics. *ACM Transactions on Graphics (TOG)*, 41(4):1–14. 140

Chen, Y., Xie, T., Yuksel, C., Kaufman, D., Yang, Y., Jiang, C., and Li, M. (2023d). Multi-layer thick shells. In *ACM SIGGRAPH Conference Proceedings*, pages 1–9. 113

Chen, Z. and Brannon, R. (2002). An evaluation of the material point method. *SAND Report, SAND2002-0482,(February 2002)*. 285

Chen, Z., Qiu, X., Zhang, X., and Lian, Y. (2015). Improved coupling of finite element method with material point method based on a particle-to-surface contact algorithm. *Computer Methods in Applied Mechanics and Engineering*, 293:1–19. 75, 114

Chen, Z., Wang, Y., Wang, F., Wang, Z., and Liu, H. (2024d). V3d: Video diffusion models are effective 3d generators. *arXiv preprint arXiv:2403.06738*. 247

Cheng, H. K., Oh, S. W., Price, B., Lee, J.-Y., and Schwing, A. (2024). Putting the object back into video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3151–3161. 232, 243

Chentanez, N., Müller, M., and Macklin, M. (2016). Real-time simulation of large elasto-plastic deformation with shape matching. In *Symposium on Computer Animation*, pages 159–167. 163

Cheon, Y.-J. and Kim, H.-G. (2018). An efficient contact algorithm for the interaction of material particles with finite elements. *Computer Methods in Applied Mechanics and Engineering*, 335:631–659. 75

Choi, K.-J. and Ko, H.-S. (2005a). Session details: Advanced topics on clothing simulation and animation. In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, New York, NY, USA. Association for Computing Machinery. 138

Choi, K.-J. and Ko, H.-S. (2005b). Stable but responsive cloth. In *ACM SIGGRAPH 2005 Courses*, pages 1–es. 139

Choi, M. G. and Ko, H.-S. (2005c). Modal warping: Real-time simulation of large rotational deformation and manipulation. *IEEE Trans. on Visualization and Computer Graphics*, 11(1):91–101. 140

Christiansen, A. N., Nobel-Jørgensen, M., Aage, N., Sigmund, O., and Bærentzen, J. A. (2014). Topology optimization using an explicit interface representation. *Structural and Multidisciplinary Optimization*, 49(3):387–399. 282

Chu, M., Liu, L., Zheng, Q., Franz, E., Seidel, H.-P., Theobalt, C., and Zayer, R. (2022). Physics informed neural fields for smoke reconstruction with sparse data. *ACM Transactions on Graphics (TOG)*, 41(4):1–14. 189

Cirio, G., Lopez-Moreno, J., Miraut, D., and Otaduy, M. A. (2014). Yarn-level simulation of woven cloth. *ACM Transactions on Graphics (TOG)*, 33(6):1–11. 139

Clavet, S., Beaudoin, P., and Poulin, P. (2005). Particle-based viscoelastic fluid simulation. In *Proceedings of the 2005 Symp. Computer animation*, pages 219–228. 19, 163

Courant, R., Friedrichs, K., and Lewy, H. (1967). On the partial difference equations of mathematical physics. *IBM journal of Research and Development*, 11(2):215–234. 70

Cummins, S. and Brackbill, J. (2002). An implicit particle-in-cell method for granular materials. *Journal of Computational Physics*, 180(2):506–548. 74, 76

Daviet, G. and Bertails-Descoubes, F. (2016). A semi-implicit material point method for the continuum simulation of granular materials. *ACM Trans. on Graph.*, 35(4). 19, 113, 163

de Vaucorbeil, A. and Nguyen, V. P. (2021). Modelling contacts with a total lagrangian material point method. *Computer Methods in Applied Mechanics and Engineering*, 373:113503. 73

de Vaucorbeil, A., Nguyen, V. P., and Hutchinson, C. R. (2020). A total-lagrangian material point method for solid mechanics problems involving large deformations. *Computer Methods in Applied Mechanics and Engineering*, 360:112783. 73, 299, 328

de Vaucorbeil, A., Nguyen, V. P., Sinaie, S., and Wu, J. Y. (2019). Material point method after 25 years: theory, implementation and applications. *Submitted to Advances in Applied Mechanics*, page 1. 73, 105, 283

De Vaucorbeil, A., Nguyen, V. P., Sinaie, S., and Wu, J. Y. (2020). Material point method after 25 years: theory, implementation, and applications. *Advances in applied mechanics*, 53:185–398. 166

Deng, N., He, Z., Ye, J., Duinkharjav, B., Chakravarthula, P., Yang, X., and Sun, Q. (2022). Fov-nerf: Foveated neural radiance fields for virtual reality. *IEEE Transactions on Visualization and Computer Graphics*, 28(11):3854–3864. 228, 230

Deul, C., Charrier, P., and Bender, J. (2016). Position-based rigid-body dynamics. *Computer Animation and Virtual Worlds*, 27(2):103–112. 241

Dhariwal, P. and Nichol, A. (2021). Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794. 249

Dinev, D., Liu, T., and Kavan, L. (2018). Stabilizing integrators for real-time physics. *ACM Trans. on Graph.*, 37(1):1–19. 21

Drucker, D. C. (1950). Some implications of work hardening and ideal plasticity. *Quarterly of Applied Mathematics*, 7(4):411–418. 54, 55

Drucker, D. C. and Prager, W. (1952). Soil mechanics and plastic analysis or limit design. *Quarterly of Applied Mathematics*, 10:157–165. 160

Du, T., Wu, K., Ma, P., Wah, S., Spielberg, A., Rus, D., and Matusik, W. (2021). DiffPD: Differentiable projective dynamics with contact. *arXiv:2101.05917*. 190, 251, 252

Duisterhof, B. P., Mandi, Z., Yao, Y., Liu, J.-W., Shou, M. Z., Song, S., and Ichnowski, J. (2023). Md-splatting: Learning metric deformation from 4d gaussians in highly deformable scenes. *arXiv preprint arXiv:2312.00583*. 227

Eckert, M.-L., Um, K., and Thuerey, N. (2019). Scalarflow: a large-scale volumetric data set of real-world scalar transport flows for computer animation and machine learning. *ACM Transactions on Graphics (TOG)*, 38(6):1–16. 54

Falanga, D., Kleber, K., Mintchev, S., Floreano, D., and Scaramuzza, D. (2018). The foldable drone: A morphing quadrotor that can squeeze and fly. *IEEE Robotics and Automation Letters*, 4(2):209–216. 318

Falkenstein, M., Jones, B., Levine, J. A., Shinar, T., and Bargteil, A. W. (2017). Reclustering for large plasticity in clustered shape matching. In *Proceedings of the Tenth International Conference on Motion in Games*, pages 1–6. 20, 163

Fan, L., Chitalu, F. M., and Komura, T. (2022). Simulating brittle fracture with material points. *ACM Transactions on Graphics (TOG)*, 41(5):1–20. 113

Fang, J., Wang, J., Zhang, X., Xie, L., and Tian, Q. (2023). Gaussianeditor: Editing 3d gaussians delicately with text instructions. *arXiv preprint arXiv:2311.16037*. 227

Fang, Y., Hu, Y., Hu, S.-M., and Jiang, C. (2018). A temporally adaptive material point method with regional time stepping. In *Computer graphics forum*, volume 37, pages 195–204. Wiley Online Library. 114

Fang, Y., Li, M., Gao, M., and Jiang, C. (2019). Silly rubber: an implicit material point method for simulating non-equilibrated viscoelastic and elastoplastic solids. *ACM Trans. on Graph.*, 38(4):1–13. 16, 19, 37, 113, 163, 185, 285

Fang, Y., Li, M., Jiang, C., and Kaufman, D. M. (2021). Guaranteed globally injective 3d deformation processing. *ACM Transactions on Graphics*, 40(4). 270

Fang, Y., Liu, J., Zhang, M., Zhang, J., Ma, Y., Li, M., Hu, Y., Jiang, C., and Liu, T. (2022). Complex locomotion skill learning via differentiable physics. *arXiv preprint arXiv:2206.02341*. 190

Faure, F., Gilles, B., Bousquet, G., and Pai, D. K. (2011). Sparse meshless models of complex deformable solids. In *ACM Trans. Graph. (TOG)*, volume 30, page 73. ACM. 140

Fei, Y., Batty, C., Grinspun, E., and Zheng, C. (2018). A multi-scale model for simulating liquid-fabric interactions. *ACM Transactions on Graphics (TOG)*, 37(4):1–16. 114

Fei, Y., Batty, C., Grinspun, E., and Zheng, C. (2019). A multi-scale model for coupling strands with shear-dependent liquid. *ACM Trans. on Graph.*, 38(6):1–20. 19, 114, 163

Fei, Y., Guo, Q., Wu, R., Huang, L., and Gao, M. (2021a). Revisiting integration in the material point method: a scheme for easier separation and less dissipation. *ACM Transactions on Graphics (TOG)*, 40(4):1–16. 160

Fei, Y., Huang, Y., and Gao, M. (2021b). Principles towards real-time simulation of material point method on modern gpus. *arXiv preprint arXiv:2111.00699*. 114

Fei, Y., Maia, H. T., Batty, C., Zheng, C., and Grinspun, E. (2017). A multi-scale model for simulating liquid-hair interactions. *ACM Trans. Graph.*, 36(4):1–17. 114

Feng, X., Huang, W., Xu, W., and Wang, H. (2022). Learning-based bending stiffness parameter estimation by a drape tester. *ACM Transactions on Graphics (TOG)*, 41(6):1–16. 139

Feng, Y., Feng, X., Shang, Y., Jiang, Y., Yu, C., Zong, Z., Shao, T., Wu, H., Zhou, K., Jiang, C., et al. (2024). Gaussian splashing: Dynamic fluid synthesis with gaussian splatting. *arXiv preprint arXiv:2401.15318*. 246

Feng, Y., Shang, Y., Li, X., Shao, T., Jiang, C., and Yang, Y. (2023). Pie-nerf: Physics-based interactive elastodynamics with nerf. *arXiv preprint arXiv:2311.13099*. 225, 228

Ferguson, Z., Li, M., Schneider, T., Gil-Ureta, F., Langlois, T., Jiang, C., Zorin, D., Kaufman, D. M., and Panozzo, D. (2021a). Intersection-free rigid body dynamics. *ACM Transactions on Graphics*, 40(4). 21, 57, 140

Ferguson, Z., Li, M., Schneider, T., Gil-Ureta, F., Langlois, T., Jiang, C., Zorin, D., Kaufman, D. M., and Panozzo, D. (2021b). Intersection-free rigid body dynamics. *ACM Transactions on Graphics (SIGGRAPH)*, 40(4). 76

Floreano, D., Mintchev, S., and Shintake, J. (2017). Foldable drones: from biology to technology. In *Bioinspiration, Biomimetics, and Bioreplication 2017*, volume 10162, page 1016203. International Society for Optics and Photonics. 318

Fratarcangeli, M., Tibaldo, V., and Pellacini, F. (2016). Vivace: A practical gauss-seidel method for stable soft body dynamics. *ACM Transactions on Graphics (TOG)*, 35(6):1–9. 150

Freeman, C. D., Frey, E., Raichuk, A., Girgin, S., Mordatch, I., and Bachem, O. (2021). Brax–a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*. 251

Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., and Kanazawa, A. (2022). Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510. 207

Frâncu, M. and Moldoveanu, F. (2017). Unified simulation of rigid and flexible bodies using position based dynamics. In Jaillet, F. and Zara, F., editors, *Workshop on Virtual Reality Interaction and Physical Simulation*. The Eurographics Association. 162

Fulton, L., Modi, V., Duvenaud, D., Levin, D. I., and Jacobson, A. (2019). Latent-space dynamics for reduced deformable simulation. In *Computer graphics forum*, volume 38, pages 379–391. Wiley Online Library. 140

Gabbard, J. L., Hix, D., and Swan, J. E. (1999). User-centered design and evaluation of virtual environments. *IEEE computer Graphics and Applications*, 19(6):51–59. 231

Gan, C., Schwartz, J., Alter, S., Schrimpf, M., Traer, J., De Freitas, J., Kubilius, J., Bhandwaldar, A., Haber, N., Sano, M., et al. (2020). Threedworld: A platform for interactive multi-modal physical simulation. *arXiv preprint arXiv:2007.04954*. 54

Gan, Y., Sun, Z., Chen, Z., Zhang, X., and Liu, Y. (2018). Enhancement of the material point method using b-spline basis functions. *International Journal for numerical methods in engineering*, 113(3):411–431. 73

Gao, L., Sun, J.-M., Mo, K., Lai, Y.-K., Guibas, L. J., and Yang, J. (2023). Scenehgn: Hierarchical graph networks for 3d indoor scene generation with fine-grained geometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 227

Gao, M., Pradhana, A., Han, X., Guo, Q., Kot, G., Sifakis, E., and Jiang, C. (2018a). Animating fluid sediment mixture in particle-laden flows. *ACM Transactions on Graphics (TOG)*, 37(4):1–11. 185, 285

Gao, M., Tampubolon, A. P., Jiang, C., and Sifakis, E. (2017). An adaptive generalized interpolation material point method for simulating elastoplastic materials. *ACM Transactions on Graphics (TOG)*, 36(6):1–12. 114, 164

Gao, M., Wang, X., Wu, K., Pradhana, A., Sifakis, E., Yuksel, C., and Jiang, C. (2018b). Gpu optimization of material point methods. *ACM Trans. on Graph.*, 37(6). 16, 114, 209

Gao, R., Holynski, A., Henzler, P., Brussee, A., Martin-Brualla, R., Srinivasan, P., Barron, J. T., and Poole, B. (2024). Cat3d: Create anything in 3d with multi-view diffusion models. *arXiv:2405.10314*. 247, 249

Garnier, P., Viquerat, J., Rabault, J., Larcher, A., Kuhnle, A., and Hachem, E. (2021). A review on deep reinforcement learning for fluid mechanics. *Computers & Fluids*, 225:104973. 54

Gast, T. F., Schroeder, C., Stomakhin, A., Jiang, C., and Teran, J. (2015). Optimization integrator for large time steps. *trans. on vis. and comp. graph.*, 21(10). 17, 21, 42, 44, 56, 57, 76, 113, 286, 328

Gaume, J., Gast, T., Teran, J., van Herwijnen, A., and Jiang, C. (2018). Dynamic anticrack propagation in snow. *Nature Communications*, 9(1):3047. 36, 54, 55, 67, 73, 131, 285

Gaume, J., van Herwijnen, A., Gast, T., Teran, J., and Jiang, C. (2019). Investigating the release and flow of snow avalanches at the slope-scale using a unified model based on the material point method. *Cold Regions Science and Technology*, 168:102847. 285

Ge, Y., Zhang, R., Wang, X., Tang, X., and Luo, P. (2019). Deepfashion2: A versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5337–5345. 274

Gea, H. C. and Luo, J. (2001). Topology optimization of structures with geometrical nonlinearities. *Computers & Structures*, 79(20-21):1977–1985. 314

Geilinger, M., Hahn, D., Zehnder, J., Bächer, M., Thomaszewski, B., and Coros, S. (2020). ADD: Analytically differentiable dynamics for multi-body systems with frictional contact. *ACM Transactions on Graphics (TOG)*, 39(6). 190, 251

Gerszewski, D., Bhattacharya, H., and Bargteil, A. W. (2009). A point-based method for animating elastoplastic solids. In *Symp. Computer animation*, pages 133–138. 20, 164

Giles, M. B. and Pierce, N. A. (2000). An introduction to the adjoint approach to design. *Flow, turbulence and combustion*, 65(3-4):393–415. 290

Gilles, B., Bousquet, G., Faure, F., and Pai, D. K. (2011). Frame-based elastic models. *ACM Trans. Graph. (TOG)*, 30(2):15. 140

Gingold, R. A. and Monaghan, J. J. (1977). Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly notices of the royal astronomical society*, 181(3):375–389. 282

Gingold, Y., Secord, A., Han, J. Y., Grinspun, E., and Zorin, D. (2004). A discrete model for inelastic deformation of thin shells. In *ACM SIGGRAPH/Eurographics symposium on computer animation*. Citeseer. 138

Gissler, C., Henne, A., Band, S., Peer, A., and Teschner, M. (2020). An implicit compressible sph solver for snow simulation. *ACM Trans. on Graph.*, 39(4). xv, 20, 164, 166, 168, 181, 182

Goldenthal, R., Harmon, D., Fattal, R., Bercovier, M., and Grinspun, E. (2007). Efficient simulation of inextensible cloth. In *ACM SIGGRAPH 2007 papers*, pages 49–es. 138

Goyal, S., Ruina, A., and Papadopoulos, J. (1991). Planar sliding with dry friction part 2. dynamics of motion. *Wear*, 143(2):331–352. 91

Grinspun, E., Hirani, A. N., Desbrun, M., and Schröder, P. (2003). Discrete shells. In *ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 62–67. 110, 113, 132, 138

Groen, J. P., Langelaar, M., Sigmund, O., and Ruess, M. (2016). Higher-order multi-resolution topology optimization using the finite cell method. *International Journal for Numerical Methods in Engineering*, 110(10):903–920. 287

Guan, S., Deng, H., Wang, Y., and Yang, X. (2022). Neurofluid: Fluid dynamics grounding with particle-driven neural radiance fields. In *ICML*. 189

Guédon, A. and Lepetit, V. (2023). Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. *arXiv preprint arXiv:2311.12775*. 223

Guest, J. K., Prévost, J. H., and Belytschko, T. (2004). Achieving minimum length scale in topology optimization using nodal design variables and projection functions. *International journal for numerical methods in engineering*, 61(2):238–254. 306

Guilkey, J., Harman, T., and Banerjee, B. (2007). An eulerian–lagrangian approach for simulating explosions of energetic devices. *Computers & structures*, 85(11-14):660–674. 73

Guilkey, J. and Weiss, J. (2001). An implicit time integration strategy for use with the material point method. In *Proceedings from the First MIT Conference on Computational Fluid and Solid Mechanics*, volume 2. 76

Guilkey, J. E. and Weiss, J. A. (2003). Implicit time integration for the material point method: Quantitative and algorithmic comparisons with the finite element method. *International Journal for Numerical Methods in Engineering*, 57(9):1323–1338. 74, 76, 285

Guo, X., Zhang, W., and Zhong, W. (2014). Doing topology optimization explicitly and geometrically—a new moving morphable components based framework. *Journal of Applied Mechanics*, 81(8):081009. 283, 285

Guo, Y. and Nairn, J. (2006). Three-dimensional dynamic fracture analysis using the material point method. *Computer Modeling in Engineering and Sciences*, 16(3):141. 285

Gupta, D. K., Langelaar, M., and van Keulen, F. (2018). QR-patterns: artefacts in multiresolution topology optimization. *Structural and Multidisciplinary Optimization*, 58(4):1335–1350. 287

Hahn, D., Banzet, P., Bern, J. M., and Coros, S. (2019). Real2sim: Visco-elastic parameter estimation from dynamic motion. *ACM Transactions on Graphics (TOG)*, 38(6):1–13. 251

Hale, K. S. and Stanney, K. M. (2014). *Handbook of virtual environments: Design, implementation, and applications*. CRC Press. 231

Hammerquist, C. C. and Nairn, J. A. (2017). A new method for material point method particle updates that reduces noise and enhances stability. *Computer methods in applied mechanics and engineering*, 318:724–738. 85

Han, X., Gast, T. F., Guo, Q., Wang, S., Jiang, C., and Teran, J. (2019). A hybrid material point method for frictional contact with diverse materials. *Proc. ACM Comput. Graph. Interact. Tech.*, 2(2):1–24. 114

Haque, A., Tancik, M., Efros, A. A., Holynski, A., and Kanazawa, A. (2023). Instruct-nerf2nerf: Editing 3d scenes with instructions. 241

Harlow, F. H. (1962). The particle-in-cell method for numerical solution of problems in fluid dynamics. Technical report, Los Alamos Scientific Lab., N. Mex. 73, 85, 283

Harlow, F. H. (1964). The particle-in-cell computing method for fluid dynamics. *Methods Comput. Phys.*, 3:319–343. 18

Hartley, R. and Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge university press. 187

Hauser, K. K., Shen, C., and O'Brien, J. F. (2003). Interactive deformation using modal analysis with constraints. In *Graphics Interface*, volume 3, pages 16–17. 140

He, K., Yao, K., Zhang, Q., Yu, J., Liu, L., and Xu, L. (2024a). Dresscode: Autoregressively sewing and generating garments from text guidance. *arXiv preprint arXiv:2401.16465*. 251

He, X., Li, X., Kang, D., Ye, J., Zhang, C., Chen, L., Gao, X., Zhang, H., Wu, Z., and Zhuang, H. (2024b). Magicman: Generative novel view synthesis of humans with 3d-aware diffusion and iterative refinement. *arXiv preprint arXiv:2408.14211*. 247, 249, 259

He, X., Wang, H., and Wu, E. (2017). Projective peridynamics for modeling versatile elastoplastic materials. *trans. on vis. and comp. graph.*, 24(9). 20, 164

Hegemann, J., Jiang, C., Schroeder, C., and Teran, J. M. (2013). A level set method for ductile fracture. In *Symp. Computer animation*, pages 193–201. 15, 18, 113, 163

Heiden, E., Macklin, M., Narang, Y. S., Fox, D., Garg, A., and Ramos, F. (2021). DiSECt: A Differentiable Simulation Engine for Autonomous Robotic Cutting. In *Proceedings of Robotics: Science and Systems*, Virtual. 190

Herholz, P., Stuyck, T., and Kavan, L. (2024). A mesh-based simulation framework using automatic code generation. *ACM Transactions on Graphics (TOG)*, 43(6):1–17. 252

Herschel, W. H. and Bulkley, R. (1926). Konsistenzmessungen von gummi-benzollösungen. *Kolloid-Zeitschrift*, 39:291–300. 160

Higo, Y., Oka, F., Kimoto, S., Morinaka, Y., Goto, Y., and Chen, Z. (2010). A coupled mpm-fdm analysis method for multi-phase elasto-plastic soils. *Soils and foundations*, 50(4):515–532. 74

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851. 249

Ho, J., Salimans, T., Gritsenko, A., Chan, W., Norouzi, M., and Fleet, D. J. (2022). Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646. 249

Hoetzlein, R. C. (2014). Fast fixed-radius nearest neighbors: interactive million-particle fluids. In *GPU Technology Conference*, volume 18, page 2. 171

Homel, M. and Herbold, E. (2018). Fracture and contact in the material point method: New approaches and applications. In *Advances in Computational Coupling and Contact Mechanics*, pages 289–326. World Scientific. 73

Homel, M. A., Brannon, R. M., and Guilkey, J. (2016). Controlling the onset of numerical fracture in parallelized implementations of the material point method (mpm) with convective particle domain interpolation (cpdi) domain scaling. *International Journal for Numerical Methods in Engineering*, 107(1):31–48. 74, 85

Homel, M. A. and Herbold, E. B. (2017). Field-gradient partitioning for fracture and frictional contact in the material point method. *International Journal for Numerical Methods in Engineering*, 109(7):1013–1044. 73, 285

Hu, Y., Anderson, L., Li, T.-M., Sun, Q., Carr, N., Ragan-Kelley, J., and Durand, F. (2020). Difftaichi: Differentiable programming for physical simulation. *ICLR*. 72, 190, 192, 195, 252

Hu, Y., Fang, Y., Ge, Z., Qu, Z., Zhu, Y., Pradhana, A., and Jiang, C. (2018a). A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Trans. on Graph.*, 37(4):150. 24, 73, 121, 164, 197, 208

Hu, Y., Li, T.-M., Anderson, L., Ragan-Kelley, J., and Durand, F. (2019a). Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics (TOG)*, 38(6):1–16. 209

Hu, Y., Liu, J., Spielberg, A., Tenenbaum, J. B., Freeman, W. T., Wu, J., Rus, D., and Matusik, W. (2019b). Chainqueen: A real-time differentiable physical simulator for soft robotics. In *2019 International conference on robotics and automation (ICRA)*, pages 6265–6271. IEEE. 251, 252, 321

Hu, Y., Zhou, Q., Gao, X., Jacobson, A., Zorin, D., and Panozzo, D. (2018b). Tetrahedral meshing in the wild. *ACM Trans. Graph.*, 37(4):60–1. 200

Huang, P., Zhang, X., Ma, S., and Huang, X. (2010). Contact algorithms for the material point method in impact and penetration simulation. *International Journal for Numerical Methods in Engineering*, 85(4):498–517. 96

Huang, Y.-H., Cao, Y.-P., Lai, Y.-K., Shan, Y., and Gao, L. (2023a). Nerf-texture: Texture synthesis with neural radiance fields. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–10. 227

Huang, Y.-H., Sun, Y.-T., Yang, Z., Lyu, X., Cao, Y.-P., and Qi, X. (2023b). Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. *arXiv preprint arXiv:2312.14937*. 228

Huang, Z., Hu, Y., Du, T., Zhou, S., Su, H., Tenenbaum, J. B., and Gan, C. (2021). Plasticinelab: A soft-body manipulation benchmark with differentiable physics. *arXiv preprint arXiv:2104.03311*. 65, 130, 190, 205, 251

Huang, Z., Tozoni, D. C., Gjoka, A., Ferguson, Z., Schneider, T., Panozzo, D., and Zorin, D. (2024). Differentiable solver for time-dependent deformation problems with contact. *ACM Transactions on Graphics*, 43(3):1–30. 251, 252, 253

Hughes, T. J. (2012). *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation. 73

Ihmsen, M., Orthmann, J., Solenthaler, B., Kolb, A., and Teschner, M. (2014). Sph fluids in computer graphics. 171

Inglis, T., Eckert, M.-L., Gregson, J., and Thuerey, N. (2017). Primal-dual optimization for fluids. In *Computer Graphics Forum*, volume 36. 21

Irving, G., Teran, J., and Fedkiw, R. (2004). Invertible finite elements for robust simulation of large deformation. In *Symp. Computer animation*, pages 131–140. 18, 20, 112

Irving, G., Teran, J., and Fedkiw, R. (2006). Tetrahedral and hexahedral invertible finite elements. *Graphical Models*, 68(2):66–89. 42, 112

Jambon, C., Kerbl, B., Kopanas, G., Diolatzis, S., Drettakis, G., and Leimkühler, T. (2023). Nerfshop: Interactive editing of neural radiance fields. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 6(1). 205

James, K. A. and Waisman, H. (2016). Layout design of a bi-stable cardiovascular stent using topology optimization. *Computer Methods in Applied Mechanics and Engineering*, 305:869–890. 319, 322

Jaques, M., Asenov, M., Burke, M., and Hospedales, T. (2022). Vision-based system identification and 3d keypoint discovery using dynamics constraints. In *Learning for Dynamics and Control Conference*, pages 316–329. PMLR. 187

Jaques, M., Burke, M., and Hospedales, T. (2020). Physics-as-inverse-graphics: Unsupervised physical parameter estimation from video. *ICLR*. 187

Jatavallabhula, K., Macklin, M., Golemo, F., Voleti, V., Petrini, L., Weiss, M., Considine, B., Parent-Lévesque, J., Xie, K., Erleben, K., et al. (2020). gradsim: Differentiable simulation for system identification and visuomotor control. In *International Conference on Learning Representations*. 187, 188, 190, 201, 202

Jatavallabhula, K. M., Macklin, M., Golemo, F., Voleti, V., Petrini, L., Weiss, M., Considine, B., Parent-Lévesque, J., Xie, K., Erleben, K., et al. (2021). gradsim: Differentiable simulation for system identification and visuomotor control. *arXiv preprint arXiv:2104.02646*. 251, 252

Jiang, B., Zhang, J., Hong, Y., Luo, J., Liu, L., and Bao, H. (2020a). Bcnet: Learning body and cloth shape from a single image. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16*, pages 18–35. Springer. 250, 272

Jiang, C., Gast, T., and Teran, J. (2017a). Anisotropic elastoplasticity for cloth, knit and hair frictional contact. *ACM Transactions on Graphics (TOG)*, 36(4):1–14. 56, 114, 160, 209

Jiang, C., Schroeder, C., Selle, A., Teran, J., and Stomakhin, A. (2015a). The affine particle-in-cell method. *ACM Transactions on Graphics (TOG)*, 34(4):1–10. 10, 74, 85, 121, 128, 160, 173, 188, 190, 191, 209

Jiang, C., Schroeder, C., Selle, A., Teran, J., and Stomakhin, A. (2015b). The affine particle-in-cell method. *ACM Trans. Graph.*, 34(4). 24, 39

Jiang, C., Schroeder, C., and Teran, J. (2017b). An angular momentum conserving affine-particle-in-cell method. *Journal of Computational Physics*, 338:137–164. 85

Jiang, C., Schroeder, C., Teran, J., Stomakhin, A., and Selle, A. (2016). The material point method for simulating continuum materials. In *ACM SIGGRAPH 2016 Courses*, pages 1–52. 15, 23, 39, 54, 57, 122, 144, 161, 166, 171, 211, 321, 346

Jiang, Y., Li, M., Jiang, C., and Alonso-Marroquin, F. (2020b). A hybrid material-point spheropolygon-element method for solid and granular material interaction. *International Journal for Numerical Methods in Engineering*, 121(14):3021–3047. 74

Jiang, Y., Yang, S., Qiu, H., Wu, W., Loy, C. C., and Liu, Z. (2022). Text2human: Text-driven controllable human image generation. *ACM Transactions on Graphics (TOG)*, 41(4):1–11. 275

Jiang, Y., Yu, C., Xie, T., Li, X., Feng, Y., Wang, H., Li, M., Lau, H., Gao, F., Yang, Y., et al. (2024). Vr-gs: a physical dynamics-aware interactive gaussian splatting system in virtual reality. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–1. 184

Jin, Y., Han, Y., Geng, Z., Teran, J., and Fedkiw, R. (2022). Analytically integratable zero-restlength springs for capturing dynamic modes unrepresented by quasistatic neural networks. *arXiv preprint arXiv:2201.10122*. 54

Johnson, K. L. and Johnson, K. L. (1987). *Contact mechanics*. Cambridge university press. 139

Jones, B., Martin, A., Levine, J. A., Shinar, T., and Bargteil, A. W. (2016a). Ductile fracture for clustered shape matching. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 65–70. 20, 163

Jones, B., Thuerey, N., Shinar, T., and Bargteil, A. W. (2016b). Example-based plastic deformation of rigid bodies. *ACM Trans. on Graph.*, 35(4). 18

Jones, B., Ward, S., Jallepalli, A., Perenia, J., and Bargteil, A. W. (2014). Deformation embedding for point-based elastoplastic simulation. *ACM Trans. on Graph.*, 33(2):1–9. 19, 163

Joshi, P., Meyer, M., DeRose, T., Green, B., and Sanocki, T. (2007). Harmonic coordinates for character articulation. *ACM transactions on graphics (TOG)*, 26(3):71–es. 258

Kalawsky, R. S. (1999). Vruse—a computerised diagnostic tool: for usability evaluation of virtual/synthetic environment systems. *Applied ergonomics*, 30(1):11–25. 229

Kaldor, J. M., James, D. L., and Marschner, S. (2008). Simulating knitted cloth at the yarn level. In *ACM SIGGRAPH 2008 papers*, pages 1–9. 139

Kane, C., Marsden, J. E., Ortiz, M., and West, M. (2000). Variational integrators and the newmark algorithm for conservative and dissipative mechanical systems. *International Journal for numerical methods in engineering*, 49(10):1295–1325. 76

Karamouzas, I., Sohre, N.and Narain, R., and Guy, S. (2017). Implicit crowds: Optimization integrator for robust crowd simulation. *ACM Trans. on Graph.*, 36(4). 21

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. (2021). Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440. 54

Karras, T. (2012). Maximizing parallelism in the construction of bvhs, octrees, and k-d trees. In *Proceedings of the Fourth ACM SIGGRAPH/Eurographics Conference on High-Performance Graphics*, pages 33–37. 179

Kaufman, D. M., Sueda, S., James, D. L., and Pai, D. K. (2008). Staggered projections for frictional contact in multibody systems. In *ACM SIGGRAPH Asia 2008 papers*, pages 1–11. 139

Ke, Y., Wang, K., and Chen, B. M. (2018). Design and implementation of a hybrid uav with model-based flight capabilities. *IEEE/ASME Transactions on Mechatronics*, 23(3):1114–1125. 318

Kerbl, B., Kopanas, G., Leimkühler, T., and Drettakis, G. (2023). 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (ToG)*, 42(4):1–14. 14, 205, 207, 208, 209, 210, 213, 216, 225, 227, 231, 243

Kim, T. (2020). A finite element formulation of baraff-witkin cloth. In *Computer Graphics Forum*, volume 39, pages 171–179. Wiley Online Library. 139

Kim, T., De Goes, F., and Iben, H. (2019). Anisotropic elasticity for inversion-safety and element rehabilitation. *ACM Trans. Graph.*, 38(4):1–15. 112

Kim, T. and Delaney, J. (2013). Subspace fluid re-simulation. *ACM Transactions on Graphics (TOG)*, 32(4):1–9. 140

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. 64

Klár, G., Gast, T., Pradhana, A., Fu, C., Schroeder, C., Jiang, C., and Teran, J. (2016). Drucker-prager elastoplasticity for sand animation. *ACM Transactions on Graphics (TOG)*, 35(4):1–12. 16, 19, 23, 25, 26, 27, 45, 46, 47, 54, 55, 66, 110, 113, 160, 163, 164, 174, 175, 184, 190, 209, 218, 285, 336, 338, 355, 357

Knapitsch, A., Park, J., Zhou, Q.-Y., and Koltun, V. (2017). Tanks and temples: Benchmarking large-scale scene reconstruction. volume 36, pages 1–13. ACM New York, NY, USA. 241

Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., and Hoyer, S. (2021). Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21). 54

Koeppe, A., Bamer, F., Selzer, M., Nestler, B., and Markert, B. (2022). Explainable artificial intelligence for mechanics: physics-explaining neural networks for constitutive models. *Front. Mater. 8: 824958. doi: 10.3389/fmats*. 56

Korosteleva, M. and Lee, S.-H. (2022). Neuraltailor: Reconstructing sewing pattern structures from 3d point clouds of garments. *ACM Transactions on Graphics (TOG)*, 41(4):1–16. 250, 274, 275

Korosteleva, M. and Sorkine-Hornung, O. (2023). Garmentcode: Programming parametric sewing patterns. *ACM Transactions on Graphics (TOG)*, 42(6):1–15. 251

Kugelstadt, T., Longva, A., Thuerey, N., and Bender, J. (2019). Implicit density projection for volume conserving liquids. *IEEE Transactions on Visualization and Computer Graphics*, 27(4):2385–2395. 173

Kularathna, S. and Soga, K. (2017). Implicit formulation of material point method for analysis of incompressible materials. *Computer Methods in Applied Mechanics and Engineering*, 313:673–686. 285

Labs, B. F. (2023). Flux. `https://github.com/black-forest-labs/flux`. 270, 275

Laine, S., Hellsten, J., Karras, T., Seol, Y., Lehtinen, J., and Aila, T. (2020). Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics*, 39(6). 260

Lan, L., Kaufman, D., Li, M., Jiang, C., and Yang, Y. (2022a). Affine body dynamics: fast, stable and intersection-free simulation of stiff materials. *ACM Trans. Graph.*, 41(4):1–14. 113, 119, 128, 140

Lan, L., Kaufman, D. M., Li, M., Jiang, C., and Yang, Y. (2022b). Affine body dynamics: Fast, stable & intersection-free simulation of stiff materials. *ACM Trans. Graph. (SIGGRAPH)*, 41(4). 57

Lan, L., Li, M., Jiang, C., Wang, H., and Yang, Y. (2023). Second-order stencil descent for interior-point hyperelasticity. *ACM Transactions on Graphics (TOG)*. 140, 154

Lan, L., Ma, G., Yang, Y., Zheng, C., Li, M., and Jiang, C. (2022c). Penetration-free projective dynamics on the gpu. *ACM Transactions on Graphics (TOG)*, 41(4):1–16. 140, 150

Lan, L., Yang, Y., Kaufman, D., Yao, J., Li, M., and Jiang, C. (2021a). Medial ipc: accelerated incremental potential contact with medial elastics. *ACM Trans. on Graph.*, 40(4):1–16. 21, 57, 140

Lan, L., Yang, Y., Kaufman, D. M., Yao, J., Li, M., and Jiang, C. (2021b). Medial IPC: Accelerated incremental potential contact with medial elastics. *ACM Transactions on Graphics (SIGGRAPH)*, 40(4). 76

Larese, A., Iaconeta, I., Chandra, B., and Singer, V. (2019). Implicit mpm and coupled mpm-fem in geomechanics. *Computational mechanics*, 175:226–232. 76

Lee, K. and Carlberg, K. T. (2020). Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404:108973. 140

Lei, X., Liu, C., Du, Z., Zhang, W., and Guo, X. (2019). Machine learning-driven real-time topology optimization under moving morphable component-based framework. *Journal of Applied Mechanics*, 86(1). 283

Li, B., Li, X., Jiang, Y., Xie, T., Gao, F., Wang, H., Yang, Y., and Jiang, C. (2024a). Garmentdreamer: 3dgs guided garment synthesis with diverse geometry and texture details. *arXiv preprint arXiv:2405.12420*. 271

Li, C., Li, S., Zhao, Y., Zhu, W., and Lin, Y. (2022a). Rt-nerf: Real-time on-device neural radiance fields towards immersive ar/vr rendering. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pages 1–9. 228

Li, J., Liu, T., and Kavan, L. (2018). Laplacian damping for projective dynamics. In *Proceedings of the 14th Workshop on Virtual Reality Interactions and Physical Simulations*, pages 29–36. 21

Li, K., Rolff, T., Bacher, R., and Steinicke, F. (2023a). Realitygit: Cross reality version control of r&d optical workbench. In *2023 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pages 807–808. IEEE. 228

Li, K., Rolff, T., Schmidt, S., Bacher, R., Frintrop, S., Leemans, W., and Steinicke, F. (2022b). Immersive neural graphics primitives. *arXiv preprint arXiv:2211.13494*. 228

Li, K., Rolff, T., Schmidt, S., Bacher, R., Leemans, W., and Steinicke, F. (2023b). Interacting with neural radiance fields in immersive virtual reality. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–4. 228

Li, K., Schmidt, S., Rolff, T., Bacher, R., Leemans, W., and Steinicke, F. (2023c). Magic nerf lens: Interactive fusion of neural radiance fields for virtual facility inspection. *arXiv preprint arXiv:2307.09860*. 228

Li, M. (2020). *Robust and Accurate Simulation of Elastodynamics and Contact*. PhD thesis, University of Pennsylvania. 55, 57, 76

Li, M., Ferguson, Z., Schneider, T., Langlois, T., Zorin, D., Panozzo, D., Jiang, C., and Kaufman, D. (2020). Incremental potential contact: Intersection-and inversion-free, large-deformation dynamics. *ACM transactions on graphics*. 12, 17, 21, 41, 43, 57, 67, 76, 86, 87, 91, 92, 111, 113, 117, 119, 136, 140, 142, 252, 328

Li, M., Ferguson, Z., Schneider, T., Langlois, T., Zorin, D., Panozzo, D., Jiang, C., and Kaufman, D. M. (N.D.). Convergent incremental potential contact for piecewise linear boundaries. unpublished. 76, 86, 87, 90

Li, M., Gao, M., Langlois, T., Jiang, C., and Kaufman, D. M. (2019a). Decomposed optimization time integrator for large-step elastodynamics. *ACM Trans. on Graph.*, 38(4). 21, 56, 83, 113, 328

Li, M., Kaufman, D. M., and Jiang, C. (2021a). Codimensional incremental potential contact. *ACM Transactions on Graphics*, 40(4). 12, 21, 57, 76, 91, 95, 113, 118, 137, 140, 142, 154, 158, 252, 253

Li, M., Lei, Y., Gao, D., Hu, Y., and Zhang, X. (2021b). A novel material point method (mpm) based needle-tissue interaction model. *Computer Methods in Biomechanics and Biomedical Engineering*, pages 1–15. 75

Li, P., Zheng, W., Liu, Y., Yu, T., Li, Y., Qi, X., Li, M., Chi, X., Xia, S., Xue, W., et al. (2024b). Pshuman: Photorealistic single-view human reconstruction using cross-scale diffusion. *arXiv preprint arXiv:2409.10141*. 247, 249

Li, R., Dumery, C., Guillard, B., and Fua, P. (2024c). Garment recovery with shape and deformation priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1586–1595. 247, 251, 272

Li, S., Huang, Z., Du, T., Su, H., Tenenbaum, J. B., and Gan, C. (2022c). Contact points discovery for soft-body manipulations with differentiable physics. *arXiv preprint arXiv:2205.02835*. 72

Li, S. and Pan, Y. (2023). Interactive geometry editing of neural radiance fields. *arXiv preprint arXiv:2303.11537*. 227

Li, X., Cao, Y., Li, M., Yang, Y., Schroeder, C., and Jiang, C. (2022d). Plasticitynet: Learning to simulate metal, sand, and snow for optimization time integration. In *Advances in Neural Information Processing Systems*. 205

Li, X., Fang, Y., Li, M., and Jiang, C. (2021c). Bfemp: Interpenetration-free mpm–fem coupling with barrier contact. *Comp. meth. applied mech. eng.* 21, 57, 69

Li, X., Fang, Y., Li, M., and Jiang, C. (2022e). Bfemp: Interpenetration-free mpm–fem coupling with barrier contact. *Computer Methods in Applied Mechanics and Engineering*, 390:114350. 114

Li, X., Li, M., and Jiang, C. (2022f). Energetically consistent inelasticity for optimization time integration. *ACM Trans. on Graph.*, 41(4). 55, 56, 59, 60

Li, X., Li, M., and Jiang, C. (2022g). Energetically consistent inelasticity for optimization time integration. *ACM Trans. Graph.*, 41(4):1–16. 114, 117, 163, 168, 174

Li, X., Qiao, Y.-L., Chen, P. Y., Jatavallabhula, K. M., Lin, M., Jiang, C., and Gan, C. (2023d). PAC-neRF: Physics augmented continuum neural radiance fields for geometry-agnostic system identification. In *The Eleventh International Conference on Learning Representations*. 208, 220, 221

Li, X., Qiao, Y.-L., Chen, P. Y., Jatavallabhula, K. M., Lin, M., Jiang, C., and Gan, C. (2023e). Pac-nerf: Physics augmented continuum neural radiance fields for geometry-agnostic system identification. *arXiv preprint arXiv:2303.05512*. 227, 239, 246, 251

Li, X., Sovilla, B., Jiang, C., and Gaume, J. (2021d). Three-dimensional and real-scale modeling of flow regimes in dense snow avalanches. *Landslides*, 18(10):3393–3406. 54, 55

Li, Y., Chen, H.-y., Larionov, E., Sarafianos, N., Matusik, W., and Stuyck, T. (2024d). Diffavatar: Simulation-ready garment optimization with differentiable simulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4368–4378. 250, 251, 252, 267

Li, Y., Du, T., Wu, K., Xu, J., and Matusik, W. (2022h). Diffcloth: Differentiable cloth simulation with dry frictional contact. *ACM Transactions on Graphics (TOG)*, 42(1):1–20. 251, 252

Li, Y., Li, X., Li, M., Zhu, Y., Zhu, B., and Jiang, C. (2021e). Lagrangian–eulerian multidensity topology optimization with the material point method. *International Journal for Numerical Methods in Engineering*, 122(14):3400–3424. 41, 95

Li, Y., Li, X., Li, M., Zhu, Y., Zhu, B., and Jiang, C. (2021f). Lagrangian-eulerian multi-density topology optimization with the material point method. *International Journal for Numerical Methods in Engineering*. 322

Li, Y., Lin, Z.-H., Forsyth, D., Huang, J.-B., and Wang, S. (2023f). Climatenerf: Extreme weather synthesis in neural radiance field. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3227–3238. 208

Li, Y., Sun, Y., Ma, P., Sifakis, E., Du, T., Zhu, B., and Matusik, W. (2024e). Neuralfluid: Neural fluidic system design and control with differentiable simulation. *arXiv preprint arXiv:2405.14903*. 251

Li, Y., Wu, J., Tedrake, R., Tenenbaum, J. B., and Torralba, A. (2019b). Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *International Conference on Learning Representations*. 190

Li, Z., Niklaus, S., Snavely, N., and Wang, O. (2021g). Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 189

Li, Z., Xu, Q., Ye, X., Ren, B., and Liu, L. (2023g). Difffr: Differentiable sph-based fluid-rigid coupling for rigid body control. *ACM Transactions on Graphics (TOG)*, 42(6):1–17. 251

Lian, Y., Zhang, X., and Liu, Y. (2011a). Coupling of finite element method with material point method by local multi-mesh contact method. *Computer Methods in Applied Mechanics and Engineering*, 200(47-48):3482–3494. 75, 114

Lian, Y., Zhang, X., and Liu, Y. (2012). An adaptive finite element material point method and its application in extreme deformation problems. *Computer methods in applied mechanics and engineering*, 241:275–285. 74, 285

Lian, Y., Zhang, X., Zhou, X., and Ma, Z. (2011b). A femp method and its application in modeling dynamic response of reinforced concrete subjected to impact loading. *Computer methods in applied mechanics and engineering*, 200(17-20):1659–1670. 74, 75, 114

Lian, Y.-P., Liu, Y., and Zhang, X. (2014). Coupling of membrane element with material point method for fluid–membrane interaction problems. *International Journal of Mechanics and Materials in Design*, 10(2):199–211. 75

Liang, J., Lin, M., and Koltun, V. (2019). Differentiable cloth simulation for inverse problems. *Advances in Neural Information Processing Systems*, 32. 252

Liang, Y., Yang, X., Lin, J., Li, H., Xu, X., and Chen, Y. (2023). Luciddreamer: Towards high-fidelity text-to-3d generation via interval score matching. *arXiv preprint arXiv:2311.11284*. 242

Lin, G., Feng-Lin, L., Shu-Yu, C., Kaiwen, J., Chunpeng, L., Lai, Y., and Hongbo, F. (2023a). Sketchfacenerf: Sketch-based facial generation and editing in neural radiance fields. *ACM Transactions on Graphics*. 227

Lin, J., Zeng, A., Wang, H., Zhang, L., and Li, Y. (2023b). One-stage 3d whole-body mesh recovery with component aware transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21159–21168. 259

Lin, S., Ryabtsev, A., Sengupta, S., Curless, B. L., Seitz, S. M., and Kemelmacher-Shlizerman, I. (2021). Real-time high-resolution background matting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8762–8771. 194

Lin, X., Huang, Z., Li, Y., Tenenbaum, J. B., Held, D., and Gan, C. (2022a). Diffskill: Skill abstraction from differentiable physics for deformable object manipulations with tools. *arXiv preprint arXiv:2203.17275*. 72

Lin, Z.-H., Ma, W.-C., Hsu, H.-Y., Wang, Y.-C. F., and Wang, S. (2022b). Neurmips: Neural mixture of planar experts for view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15702–15712. 207

Lindell, D. B., Martel, J. N., and Wetzstein, G. (2021). Autoint: Automatic integration for fast neural volume rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14556–14565. 227

Liu, C., Sun, Q., and Yang, Y. (2017a). Multi-scale modelling of granular pile collapse by using material point method and discrete element method. *Procedia Engineering*, 175:29–35. 74

Liu, C. and Sun, W. (2020). ILS-MPM: An implicit level-set-based material point method for frictional particulate contact mechanics of deformable particles. *Computer Methods in Applied Mechanics and Engineering*, 369:113168. 100, 101

Liu, H., Hu, Y., Zhu, B., Matusik, W., and Sifakis, E. (2018). Narrow-band topology optimization on a sparsely populated grid. In *SIGGRAPH Asia 2018 Technical Papers*, page 251. ACM. 282, 290, 291

Liu, L., Gu, J., Lin, K. Z., Chua, T.-S., and Theobalt, C. (2020). Neural sparse voxel fields. *NeurIPS*. 227

Liu, L., Xu, X., Lin, Z., Liang, J., and Yan, S. (2023a). Towards garment sewing pattern reconstruction from a single image. *ACM Transactions on Graphics (TOG)*, 42(6):1–15. 247, 251, 257, 272, 274, 275

Liu, L., Zhang, L., Xu, Y., Gotsman, C., and Gortler, S. J. (2008). A local/global approach to mesh parameterization. In *Computer Graphics Forum*, volume 27, pages 1495–1504. Wiley Online Library. 144

Liu, R., Wu, R., Van Hoorick, B., Tokmakov, P., Zakharov, S., and Vondrick, C. (2023b). Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9298–9309. 247

Liu, R., Xiang, J., Zhao, B., Zhang, R., Yu, J., and Zheng, C. (2023c). Neural impostor: Editing neural radiance fields with explicit shape manipulation. *arXiv preprint arXiv:2310.05391*. 208

Liu, T., Bouaziz, S., and Kavan, L. (2017b). Quasi-newton methods for real-time simulation of hyperelastic materials. *ACM Trans. on Graph.*, 36(3). 21, 148, 286

Liu, W. K., Jun, S., and Zhang, Y. F. (1995). Reproducing kernel particle methods. *International journal for numerical methods in fluids*, 20(8-9):1081–1106. 166

Lombardi, S., Simon, T., Schwartz, G., Zollhoefer, M., Sheikh, Y., and Saragih, J. (2021). Mixture of volumetric primitives for efficient neural rendering. *ACM Transactions on Graphics (ToG)*, 40(4):1–13. 227

Long, C., Zhang, D., Bronkhorst, C., and Gray III, G. (2016). Representing ductile damage with the dual domain material point method. *Computer Methods in Applied Mechanics and Engineering*, 300:611–627. 85, 285

Long, C. C., Moutsanidis, G., Bazilevs, Y., and Zhang, D. Z. (2019). Using nurbs as shape functions for mpm. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States). 85, 285

Long, X., Guo, Y.-C., Lin, C., Liu, Y., Dou, Z., Liu, L., Ma, Y., Zhang, S.-H., Habermann, M., Theobalt, C., et al. (2024). Wonder3d: Single image to 3d using cross-domain diffusion. In *Computer Vision and Pattern Recognition (CVPR)*, pages 9970–9980. 249

Lorensen, W. E. and Cline, H. E. (1987). Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169. 233

Love, E. and Sulsky, D. L. (2006). An unconditionally stable, energy–momentum consistent implementation of the material-point method. *Computer Methods in Applied Mechanics and Engineering*, 195(33-36):3903–3925. 76

Luiten, J., Kopanas, G., Leibe, B., and Ramanan, D. (2023). Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *arXiv preprint arXiv:2308.09713*. 208, 221

Luo, R., Shao, T., Wang, H., Xu, W., Chen, X., Zhou, K., and Yang, Y. (2018). Nnwarp: Neural network-based nonlinear deformation. *IEEE transactions on visualization and computer graphics*, 26(4):1745–1759. 54

Luo, R., Xu, W., Shao, T., Xu, H., and Yang, Y. (2019). Accelerated complex-step finite difference for expedient deformable simulation. *ACM Transactions on Graphics (TOG)*, 38(6):1–16. 252

Luo, Z., Liu, H., Li, C., Du, W., Jin, Z., Sun, W., Nie, Y., Chen, W., and Han, X. (2024). Garverselod: High-fidelity 3d garment reconstruction from a single in-the-wild image using a dataset with levels of details. *ACM Transactions on Graphics (TOG)*, 43(6):1–12. 250

Luo, Z., Wang, M. Y., Wang, S., and Wei, P. (2008). A level set-based parameterization method for structural shape and topology optimization. *International Journal for Numerical Methods in Engineering*, 76(1):1–26. 282

Lusch, B., Kutz, J. N., and Brunton, S. L. (2018). Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1):4950. 141

Ly, M., Jouve, J., Boissieux, L., and Bertails-Descoubes, F. (2020). Projective dynamics with dry frictional contact. *ACM Transactions on Graphics (TOG)*, 39(4):57–1. 139

Ma, P., Du, T., Tenenbaum, J. B., Matusik, W., and Gan, C. (2021). Risp: Rendering-invariant state predictor with differentiable simulation and rendering for cross-domain parameter estimation. In *International Conference on Learning Representations*. 72, 187, 188, 190

Ma, S., Zhang, X., Lian, Y., and Zhou, X. (2009). Simulation of high explosive explosion using adaptive material point method. *Computer Modeling in Engineering and Sciences (CMES)*, 39(2):101. 285

Macklin, M. (2022). Warp: A high-performance python framework for gpu simulation and graphics. In *NVIDIA GPU Technology Conference (GTC)*. 252, 271

Macklin, M., Erleben, K., Müller, M., Chentanez, N., Jeschke, S., and Kim, T.-Y. (2020). Primal/dual descent methods for dynamics. In *Computer Graphics Forum*, volume 39, pages 89–100. Wiley Online Library. 139, 182

Macklin, M. and Müller, M. (2013). Position based fluids. *ACM Trans. Graph.*, 32(4). 160, 162, 171, 172, 183

Macklin, M., Müller, M., and Chentanez, N. (2016). Xpbd: position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games*, MIG '16, page 49–54, New York, NY, USA. Association for Computing Machinery. 11, 159, 162, 164, 172, 175, 225, 234, 252, 271

Macklin, M., Müller, M., Chentanez, N., and Kim, T.-Y. (2014). Unified particle physics for real-time applications. *ACM Transactions on Graphics (TOG)*, 33(4):1–12. 162, 163, 172, 173, 175

Macklin, M. and Müller, M. (2021). A constraint-based formulation of stable neo-hookean materials. In *Motion, Interaction and Games*, pages 1–7. 159, 162, 165, 166

Macklin, M., Storey, K., Lu, M., Terdiman, P., Chentanez, N., Jeschke, S., and Müller, M. (2019). Small steps in physics simulation. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 1–7. 150, 162, 177, 180

Margossian, C. C. (2019). A review of automatic differentiation and its efficient implementation. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 9(4):e1305. 252

Martin, S., Kaufmann, P., Botsch, M., Grinspun, E., and Gross, M. (2010). Unified simulation of elastic rods, shells, and solids. In *ACM Trans. Graph. (TOG)*, volume 29, page 39. ACM. 140

Maute, K. and Sigmund, O. (2013). Topology optimization approaches: A comparative review. *Structural and Multidisciplinary Optimization*, 6. 285

Mazhar, H., Heyn, T., Negrut, D., and Tasora, A. (2015). Using nesterov's method to accelerate multibody dynamics with friction and contact. *ACM Transactions on Graphics (TOG)*, 34(3):1–14. 139

McKIVER, W. J. and Dritschel, D. G. (2003). The motion of a fluid ellipsoid in a general linear background flow. *Journal of Fluid Mechanics*, 474:147–173. 214

McNamara, A., Treuille, A., Popović, Z., and Stam, J. (2004). Fluid control using the adjoint method. *ACM Transactions On Graphics (TOG)*, 23(3):449–456. 251

Meng, X., Chen, W., and Yang, B. (2023). Neat: Learning neural implicit surfaces with arbitrary topologies from multi-view images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 248–258. 227

Michell, A. (1904). LVIII. the limits of economy of material in frame-structures. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 8(47):589–597. 308

Miguel, E., Bradley, D., Thomaszewski, B., Bickel, B., Matusik, W., Otaduy, M. A., and Marschner, S. (2012). Data-driven estimation of cloth simulation models. In *Computer Graphics Forum*, volume 31, pages 519–528. Wiley Online Library. 139

Miguel, E., Tamstorf, R., Bradley, D., Schvartzman, S. C., Thomaszewski, B., Bickel, B., Matusik, W., Marschner, S., and Otaduy, M. A. (2013). Modeling and estimation of internal friction in cloth. *ACM Transactions on Graphics (TOG)*, 32(6):1–10. 139

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*. 14, 188, 189, 191

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2021). Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106. 205, 207, 210, 223, 227

Mises, R. v. (1913a). Mechanik der festen körper im plastisch-deformablen zustand. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, 1913:582–592. 54, 56, 67

Mises, R. v. (1913b). Mechanik der festen körper im plastisch- deformablen zustand. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, 1913:582–592. 160

Molino, N., Bao, Z., and Fedkiw, R. (2004). A virtual node algorithm for changing mesh topology during simulation. *ACM Trans. on Graph.*, 23(3). 15, 18

Moon, G., Nam, H., Shiratori, T., and Lee, K. M. (2022). 3d clothed human reconstruction in the wild. In *European conference on computer vision*, pages 184–200. Springer. 250, 272

Moreau, J. J. (2011). On unilateral constraints, friction and plasticity. In *New variational techniques in mathematical physics*, pages 171–322. Springer. 91

Morton, S. and Papanikolopoulos, N. (2017). A small hybrid ground-air vehicle concept. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5149–5154. IEEE. 318

Mozaffar, M., Bostanabad, R., Chen, W., Ehmann, K., Cao, J., and Bessa, M. (2019). Deep learning predicts path-dependent plasticity. *Proceedings of the National Academy of Sciences*, 116(52):26414–26420. 56

Müller, M., Chentanez, N., Kim, T.-Y., and Macklin, M. (2015). Strain based dynamics. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '14, page 149–157, Goslar, DEU. Eurographics Association. 162

Müller, M., Chentanez, N., and Macklin, M. (2016). Simulating visual geometry. In *Proceedings of the 9th International Conference on Motion in Games*, pages 31–38. 206

Müller, M., Chentanez, N., and Macklin, M. (2016). Simulating visual geometry. In *Proceedings of the 9th International Conference on Motion in Games*, MIG '16, page 31–38, New York, NY, USA. Association for Computing Machinery. 230

Müller, M., Heidelberger, B., Hennix, M., and Ratcliff, J. (2007). Position based dynamics. *J. Visual Communication and Image Repres.*, 18(2):109–118. 20, 159, 162, 172, 252

Müller, M., Heidelberger, B., Teschner, M., and Gross, M. (2005). Meshless deformations based on shape matching. *ACM Trans. on Graph.*, 24(3). 20

Müller, M., Keiser, R., Nealen, A., Pauly, M., Gross, M., and Alexa, M. (2004). Point based animation of elastic, plastic and melting objects. In *Proceedings of the 2004 Symp. Computer animation*, pages 141–151. 19, 163

Müller, M., Kim, T., and Chentanez, N. (2012). Fast simulation of inextensible hair and fur. 159, 162

Müller, M., Macklin, M., Chentanez, N., Jeschke, S., and Kim, T.-Y. (2020). Detailed rigid body simulation with extended position based dynamics. *Computer Graphics Forum*, 39(8):101–112. 159, 162

Müller, T., Evans, A., Schied, C., and Keller, A. (2022). Instant neural graphics primitives with a multiresolution hash encoding. *arXiv preprint arXiv:2201.05989*. 207, 216, 227, 241

Museth, K. (2013). Vdb: High-resolution sparse volumes with dynamic topology. *ACM Trans. Graph.*, 32(3). 233

Nair, A. and Roy, S. (2012). Implicit time integration in the generalized interpolation material point method for finite deformation hyperelasticity. *Mechanics of Advanced Materials and Structures*, 19(6):465–473. 76, 285

Nairn, J. A. (2003). Material point method calculations with explicit cracks. *Computer Modeling in Engineering and Sciences*, 4(6):649–664. 73

Nakamura, K., Matsumura, S., and Mizutani, T. (2021). Particle-to-surface frictional contact algorithm for material point method using weighted least squares. *Computers and Geotechnics*, 134:104069. 77

Narain, R., Golas, A., and Lin, M. C. (2010). Free-flowing granular materials with two-way solid coupling. In *ACM SIGGRAPH Asia 2010 papers*, pages 1–10. 21

Narain, R., Overby, M., and Brown, G. (2016). Admm $\supseteq$ projective dynamics: fast simulation of general constitutive models. In *Symp. Comput. Animat.*, volume 1, page 2016. 21

Narain, R., Pfaff, T., and O'Brien, J. F. (2013). Folding and crumpling adaptive sheets. *ACM Trans. on Graph.*, 32(4):1–8. 53

Narain, R., Samii, A., and O'Brien, J. F. (2012). Adaptive anisotropic remeshing for cloth simulation. *ACM Trans. on Graph.*, 31(6):1–10. 21

Naumann, U. (2011). *The art of differentiating computer programs: an introduction to algorithmic differentiation*. SIAM. 252

Nguyen, T. H., Paulino, G. H., Song, J., and Le, C. H. (2009). A computational paradigm for multiresolution topology optimization (MTOP). *Structural and Multidisciplinary Optimization*, 41(4):525–539. 287

Nguyen, V. P., Nguyen, C. T., Rabczuk, T., and Natarajan, S. (2017). On a family of convected particle domain interpolations in the material point method. *Finite Elements in Analysis and Design*, 126:50–64. 85

Nguyen, V. P. and Nguyen, G. D. (2016). A voronoi cell material point method for large deformation solid mechanics problems. In *Applied Mechanics and Materials*, volume 846, pages 108–113. Trans Tech Publ. 285

Nichol, A., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., Sutskever, I., and Chen, M. (2021). Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*. 249

Niemeyer, M. and Geiger, A. (2021). Campari: Camera-aware decomposed generative neural radiance fields. In *2021 International Conference on 3D Vision (3DV)*, pages 951–961. IEEE. 227

Nocedal, J. and Wright, S. J. (1999). *Numerical optimization*. Springer. 94, 290, 322

Numerow, L., Li, Y., Coros, S., and Thomaszewski, B. (2024). Differentiable voronoi diagrams for simulation of cell-based mechanical systems. *ACM Transactions on Graphics (TOG)*, 43(4):1–11. 251

O'Brien, J. F., Bargteil, A. W., and Hodgins, J. K. (2002). Graphical modeling and animation of ductile fracture. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 291–294. 15, 17, 19, 110

O'brien, J. F. and Hodgins, J. K. (1999). Graphical modeling and animation of brittle fracture. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 137–146. 113

Oechsle, M., Peng, S., and Geiger, A. (2021). Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5589–5599. 227

Ortiz, M. and Stainier, L. (1999). The variational formulation of viscoplastic constitutive updates. *Comp. meth. applied mech. eng.*, 171(3-4):419–444. 17, 76, 80, 83

Osher, S. and Sethian, J. A. (1988). Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of computational physics*, 79(1):12–49. 282

Overby, M., Brown, G. E., Li, J., and Narain, R. (2017). ADMM ⊇ projective dynamics: Fast simulation of hyperelastic models with dynamic constraints. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 23(10):2222–2234. 113

Overvelde, J. T. (2012). *The Moving Node Approach in Topology Optimization*. PhD thesis, Delft University of Technology. 283

Panetta, J., Isvoranu, F., Chen, T., Siéfert, E., Roman, B., and Pauly, M. (2021). Computational inverse design of surface-based inflatables. *ACM Transactions on Graphics (TOG)*, 40(4):1–14. 251

Park, K., Sinha, U., Barron, J. T., Bouaziz, S., Goldman, D. B., Seitz, S. M., and Martin-Brualla, R. (2021). Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5865–5874. 188, 189, 208, 225, 228

Park, Y. and Agrawal, P. (2024). Using apple vision pro to train and control robots. 185

Pavlakos, G., Choutas, V., Ghorbani, N., Bolkart, T., Osman, A. A., Tzionas, D., and Black, M. J. (2019). Expressive body capture: 3d hands, face, and body from a single image. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10975–10985. 259

Pedersen, C. B., Buhl, T., and Sigmund, O. (2001). Topology synthesis of large-displacement compliant mechanisms. *International Journal for numerical methods in engineering*, 50(12):2683–2705. 325

Peng, S., Dong, J., Wang, Q., Zhang, S., Shuai, Q., Zhou, X., and Bao, H. (2021). Animatable neural radiance fields for modeling dynamic human bodies. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14314–14323. 227

Peng, Y., Yan, Y., Liu, S., Cheng, Y., Guan, S., Pan, B., Zhai, G., and Yang, X. (2022). Cagenerf: Cage-based neural radiance field for generalized 3d deformation and animation. *Advances in Neural Information Processing Systems*, 35:31402–31415. 205, 208

Pentland, A. and Williams, J. (1989). Good vibrations: Modal dynamics for graphics and animation. In *SIGGRAPH Comput. Graph.*, volume 23. ACM. 140

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. W. (2020). Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*. 54

Pilkington, N. (2022). Dronedeploy nerf dataset. 216

Plunder, S. and Merino-Aceituno, S. (2023). Convergence proof for first-order position-based dynamics: An efficient scheme for inequality constrained odes. *arXiv preprint arXiv:2310.01215*. 162

Prager, W. (1955). The theory of plasticity: a survey of recent achievements. *Proceedings of the Institution of Mechanical Engineers*, 169(1):41–57. 54, 55

Prasad, J. and Diaz, A. R. (2005). Synthesis of bistable periodic structures using topology optimization and a genetic algorithm. *Journal of Mechanical Design*, 128(6):1298–1306. 319

Provot, X. et al. (1995). Deformation constraints in a mass-spring model to describe rigid cloth behaviour. In *Graphics interface*, pages 147–147. Canadian Information Processing Society. 138

Pumarola, A., Corona, E., Pons-Moll, G., and Moreno-Noguer, F. (2021a). D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10318–10327. 188, 189

Pumarola, A., Corona, E., Pons-Moll, G., and Moreno-Noguer, F. (2021b). D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327. 208, 225, 228

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660. 226

Qiao, Y., Liang, J., Koltun, V., and Lin, M. (2021a). Differentiable simulation of soft multi-body systems. *Advances in Neural Information Processing Systems*, 34:17123–17135. 72, 190

Qiao, Y.-L., Gao, A., and Lin, M. C. (2022). Neuphysics: Editable neural geometry and physics from monocular videos. In *Conference on Neural Information Processing Systems (NeurIPS)*. 190, 208

Qiao, Y.-L., Gao, A., Xu, Y., Feng, Y., Huang, J.-B., and Lin, M. C. (2023). Dynamic mesh-aware radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 385–396. 208, 227

Qiao, Y.-L., Liang, J., Koltun, V., and Lin, M. C. (2021b). Efficient differentiable simulation of articulated bodies. In *International Conference on Machine Learning*, pages 8661–8671. PMLR. 251

Qiu, Y., Reeve, S. T., Li, M., Yang, Y., Slattery, S. R., and Jiang, C. (2023). A sparse distributed gigascale resolution material point method. *ACM Transactions on Graphics*, 42(2):1–21. 114, 209

Qu, Z., Li, M., De Goes, F., and Jiang, C. (2022). The power particle-in-cell method. *ACM Transactions on Graphics*, 41(4). 164

Qu, Z., Li, M., Yang, Y., Jiang, C., and De Goes, F. (2023). Power plastics: A hybrid lagrangian/eulerian solver for mesoscale inelastic flows. *ACM Transactions on Graphics (TOG)*, 42(6):1–11. 164, 165

Raafat, M. (2023). BlenderNeRF. 216, 218, 223

Radovitzky, R. and Ortiz, M. (1999). Error estimation and adaptive meshing in strongly nonlinear dynamic problems. *Comp. meth. applied mech. eng.*, 172(1-4). 17, 80

Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707. 54

Ram, D., Gast, T., Jiang, C., Schroeder, C., Stomakhin, A., Teran, J., and Kavehpour, P. (2015). A material point method for viscoelastic fluids, foams and sponges. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 157–163. 110, 113, 160, 163

Raymond, S. J., Jones, B., and Williams, J. R. (2018). A strategy to couple the material point method (mpm) and smoothed particle hydrodynamics (sph) computational techniques. *Computational Particle Mechanics*, 5(1):49–58. 74

Rebain, D., Jiang, W., Yazdani, S., Li, K., Yi, K. M., and Tagliasacchi, A. (2021). Derf: Decomposed radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14153–14161. 227

Redon, S., Kheddar, A., and Coquillart, S. (2002). Fast continuous collision detection between rigid bodies. In *Computer graphics forum*, volume 21, pages 279–287. Wiley Online Library. 139

Reimann, D., Nidadavolu, K., ul Hassan, H., Vajragupta, N., Glasmachers, T., Junker, P., and Hartmaier, A. (2019). Modeling macroscopic material behavior with machine learning algorithms trained by micromechanical simulations. *Frontiers in Materials*, 6:181. 56

Renardy, M. and Rogers, R. C. (2006). *An introduction to partial differential equations*, volume 13. Springer Science & Business Media. 252

Rojas, J., Sifakis, E., and Kavan, L. (2021). Differentiable implicit soft-body physics. *arXiv preprint arXiv:2102.05791*. 190

Rolff, T., Schmidt, S., Li, K., Steinicke, F., and Frintrop, S. (2023). Vrs-nerf: Accelerating neural radiance field rendering with variable rate shading. In *2023 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 243–252. IEEE. 228

Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M., and Aberman, K. (2023). Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22500–22510. 249

Sadeghirad, A., Brannon, R. M., and Burghardt, J. (2011). A convected particle domain interpolation technique to extend applicability of the material point method for problems involving massive deformations. *International Journal for numerical methods in Engineering*, 86(12):1435–1456. 73, 285

Sadeghirad, A., Brannon, R. M., and Guilkey, J. (2013). Second-order convected particle domain interpolation (cpdi2) with enrichment for weak discontinuities at material interfaces. *International Journal for numerical methods in Engineering*, 95(11):928–952. 85

Saharia, C., Chan, W., Chang, H., Lee, C., Ho, J., Salimans, T., Fleet, D., and Norouzi, M. (2022). Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 conference proceedings*, pages 1–10. 249

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. (2020). Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pages 8459–8468. PMLR. 53, 190

Scharff, R. B., Wu, J., Geraedts, J. M., and Wang, C. C. (2019). Reducing out-of-plane deformation of soft robotic actuators for stable grasping. In *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE. 286

Schechter, H. and Bridson, R. (2012). Ghost sph for animating water. *ACM Transactions on Graphics (TOG)*, 31(4):1–8. 172

Schenck, C. and Fox, D. (2018). Spnets: Differentiable fluid dynamics for deep neural networks. In *Conference on Robot Learning*, pages 317–335. PMLR. 251

Schönberger, J. L. and Frahm, J.-M. (2016). Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 217

Schönberger, J. L., Zheng, E., Pollefeys, M., and Frahm, J.-M. (2016). Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*. 217

Shen, S., Shao, T., Zhou, K., Jiang, C., Luo, F., and Yang, Y. (2022). Hod-net: High-order differentiable deep neural networks and applications. 54

Shen, S., Yin, Y., Shao, T., Wang, H., Jiang, C., Lan, L., and Zhou, K. (2021). High-order differentiable autoencoder for nonlinear model reduction. *arXiv preprint arXiv:2102.11026*. 54, 140, 252

Shewchuk, J. R. (2008). A two-dimensional quality mesh generator and delaunay triangulator. *Computer Science Division University of California at Berkeley, Berkeley, California, http://www. cs. cmu. edu/quake/triangle. html*, pages 94720–1776. 257

Shi, R., Chen, H., Zhang, Z., Liu, M., Xu, C., Wei, X., Chen, L., Zeng, C., and Su, H. (2023). Zero123++: a single image to consistent multi-view diffusion base model. *arXiv preprint arXiv:2310.15110*. 249

Shi, Y., Wang, P., Ye, J., Mai, L., Li, K., and Yang, X. Mvdream: Multi-view diffusion for 3d generation. In *The Twelfth International Conference on Learning Representations*. 249

Si, H. (2015). Tetgen, a delaunay-based quality tetrahedral mesh generator. *ACM Trans. Math. Softw.*, 41(2). 233

Si, Z., Zhang, G., Ben, Q., Romero, B., Xian, Z., Liu, C., and Gan, C. (2024). Difftactile: A physics-based differentiable tactile simulator for contact-rich robotic manipulation. *arXiv preprint arXiv:2403.08716*. 251

Sifakis, E. and Barbic, J. (2012). Fem simulation of 3d deformable solids: a practitioner's guide to theory, discretization and model reduction. In *Acm siggraph 2012 courses*, pages 1–50. 54, 140

Sigmund, O. (2001). A 99 line topology optimization code written in matlab. *Structural and multidisciplinary optimization*, 21(2):120–127. 282, 285, 287, 289, 291, 297

Silling, S. A. (2000). Reformulation of elasticity theory for discontinuities and long-range forces. *Journal of the Mechanics and Physics of Solids*, 48(1):175–209. 20, 164

Simo, J. C. (1992). Algorithms for static and dynamic multiplicative plasticity that preserve the classical return mapping schemes of the infinitesimal theory. *Comp. meth. applied mech. eng.*, 99(1):61–112. 22

Simo, J. C. and Hughes, T. J. (1998). *Computational inelasticity*. Springer-Verlag. 17, 22, 25, 28, 212, 340

Sinaie, S., Ngo, T. D., Nguyen, V. P., and Rabczuk, T. (2018). Validation of the material point method for the simulation of thin-walled tubes under lateral compression. *Thin-Walled Structures*, 130:32–46. 285

Smith, B., Goes, F. D., and Kim, T. (2018). Stable neo-hookean flesh simulation. *ACM Transactions on Graphics (TOG)*, 37(2):1–15. 112, 159, 286

Smith, J. and Schaefer, S. (2015). Bijective parameterization with free boundaries. *ACM Trans. on Graph.*, 34(4):1–9. 41, 301

Soga, K., Alonso, E., Yerro, A., Kumar, K., and Bandara, S. (2016). Trends in large-deformation analysis of landslide mass movements with particular emphasis on the material point method. *Géotechnique*, 66(3):248–273. 285

Song, L., Chen, A., Li, Z., Chen, Z., Chen, L., Yuan, J., Xu, Y., and Geiger, A. (2023). Nerfplayer: A streamable dynamic scene representation with decomposed neural radiance fields. *IEEE Transactions on Visualization and Computer Graphics*, 29(5):2732–2742. 228

Song, Y., Liu, Y., and Zhang, X. (2020). A non-penetration fem-mpm contact algorithm for complex fluid-structure interaction problems. *Computers & Fluids*, 213:104749. 75

Sorkine, O. and Alexa, M. (2007). As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, pages 109–116. Citeseer. 222

Sorkine, O. and Cohen-Or, D. (2004). Least-squares meshes. In *Proceedings Shape Modeling Applications, 2004.*, pages 191–199. IEEE. 226

Spencer, A. J. M. (2004). *Continuum mechanics*. Courier Corporation. 188

Sperl, G., Narain, R., and Wojtan, C. (2020). Homogenized yarn-level cloth. *ACM Transactions on Graphics (TOG)*, 39(4):48–1. 139

Stanney, K. M., Mollaghasemi, M., Reeves, L., Breaux, R., and Graeber, D. A. (2003). Usability engineering of virtual environments (ves): identifying multiple criteria that drive effective ve system design. *International Journal of Human-Computer Studies*, 58(4):447–481. 231

Steffen, M., Kirby, R., and Berzins, M. (2008). Analysis and reduction of quadrature errors in the material point method (MPM). *Int J Numer Meth Eng*, 76(6):922–948. 85, 285

Stomakhin, A., Howes, R., Schroeder, C., and Teran, J. M. (2012). Energetically consistent invertible elasticity. In *Proceedings of the 11th ACM SIGGRAPH/Eurographics conference on Computer Animation*, pages 25–32. 26, 54, 165

Stomakhin, A., Schroeder, C., Chai, L., Teran, J., and Selle, A. (2013). A material point method for snow simulation. *ACM Transactions on Graphics (TOG)*, 32(4):1–10. 16, 18, 20, 36, 110, 113, 160, 163, 164, 176, 182, 209, 211, 285

Stomakhin, A., Schroeder, C., Jiang, C., Chai, L., Teran, J., and Selle, A. (2014). Augmented mpm for phase-change and varied materials. *ACM Transactions on Graphics (TOG)*, 33(4):1–11. 113, 160, 163

Strecke, M. and Stueckler, J. (2021). Diffsdfsim: Differentiable rigid-body dynamics with implicit shapes. In *2021 international conference on 3D Vision (3DV)*, pages 96–105. IEEE. 251

Stuyck, T. and Chen, H.-y. (2023). Diffxpbd: Differentiable position-based simulation of compliant constraint dynamics. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 6(3):1–14. 251, 252

Su, H., Xue, T., Han, C., Jiang, C., and Aanjaneya, M. (2021). A unified second-order accurate in time mpm formulation for simulating viscoelastic liquids with phase change. *ACM Transactions on Graphics (TOG)*, 40(4):1–18. 113

Sulsky, D., Chen, Z., and Schreyer, H. L. (1994). A particle method for history-dependent materials. *Comp. meth. applied mech. eng.*, 118(1-2). 18, 163

Sulsky, D., Zhou, S.-J., and Schreyer, H. L. (1995). Application of a particle-in-cell method to solid mechanics. *Computer physics communications*, 87(1-2):236–252. 73, 283, 285

Sun, C., Sun, M., and Chen, H.-T. (2022). Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5459–5469. 188, 189, 191, 193, 195, 197, 207

Sun, J.-M., Wu, T., Yang, Y.-L., Lai, Y.-K., and Gao, L. (2023). Sol-nerf: Sunlight modeling for outdoor scene decomposition and relighting. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–11. 227

Sun, L., Gao, H., Pan, S., and Wang, J.-X. (2020a). Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, 361:112732. 54

Sun, Y., Shinar, T., and Schroeder, C. (2020b). Effective time step restrictions for explicit mpm simulation. In *Computer Graphics Forum*, volume 39. 46

Sutcliffe, A. G., Poullis, C., Gregoriades, A., Katsouri, I., Tzanavari, A., and Herakleous, K. (2019). Reflecting on the design process for virtual reality applications. *International Journal of Human–Computer Interaction*, 35(2):168–179. 230

Suvorov, R., Logacheva, E., Mashikhin, A., Remizova, A., Ashukha, A., Silvestrov, A., Kong, N., Goka, H., Park, K., and Lempitsky, V. (2022). Resolution-robust large mask inpainting with fourier convolutions. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2149–2159. 233, 236, 237, 243

Svanberg, K. (1987a). The method of moving asymptotes—a new method for structural optimization. *International Journal for Numerical Methods in Engineering*, 24(2):359–373. 322

Svanberg, K. (1987b). The method of moving asymptotes—a new method for structural optimization. *International journal for numerical methods in engineering*, 24(2):359–373. 291, 304

Swartz, K. E. and James, K. A. (2019). Gaussian layer connectivity parameterization: A new approach to topology optimization of multi-body mechanisms. *Computer-Aided Design*, 115:42–51. 320, 325

Swensen, D., Denison, M., Guilkey, J., Harman, T., and Goetz, R. (2006). A software framework for blast event simulation. *Report of Reaction Engineering International*, page 9. 74

Takahashi, T. and Batty, C. (2021). Frictionalmonolith: a monolithic optimization-based approach for granular flow with contact-aware rigid-body coupling. *ACM Transactions on Graphics (TOG)*, 40(6):1–20. 139

Takahashi, T., Dobashi, Y., Fujishiro, I., Nishita, T., and Lin, M. C. (2015). Implicit formulation for sph-based viscous fluids. In *Computer Graphics Forum*, volume 34, pages 493–502. Wiley Online Library. 164

Takahashi, T. and Lin, M. C. (2019a). A geometrically consistent viscous fluid solver with two-way fluid-solid coupling. In *Computer Graphics Forum*, volume 38, pages 49–58. Wiley Online Library. 178

Takahashi, T. and Lin, M. C. (2019b). Video-guided real-to-virtual parameter transfer for viscous fluids. *ACM Transactions on Graphics (TOG)*, 38(6):1–12. 190

Tampubolon, A. P., Gast, T., Klár, G., Fu, C., Teran, J., Jiang, C., and Museth, K. (2017). Multi-species simulation of porous sand and water mixtures. *ACM Trans. on Graph.*, 36(4):105. 16, 54, 55, 160, 163, 174, 185, 285

Tan, H. and Nairn, J. A. (2002). Hierarchical, adaptive, material point method for dynamic energy release rate calculations. *Computer Methods in Applied Mechanics and Engineering*, 191(19-20):2123–2137. 73

Tan, Y. H. and Chen, B. M. (2020). A morphable aerial-aquatic quadrotor with coupled symmetric thrust vectoring. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2223–2229. IEEE. 318

Tancik, M., Weber, E., Ng, E., Li, R., Yi, B., Wang, T., Kristoffersen, A., Austin, J., Salahi, K., Ahuja, A., et al. (2023). Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–12. 216

Tang, J., Ren, J., Zhou, H., Liu, Z., and Zeng, G. (2023). Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653*. 215, 249

Tang, M., Manocha, D., Otaduy, M. A., and Tong, R. (2012). Continuous penalty forces. *ACM Transactions on Graphics (TOG)*, 31(4):1–9. 139

Tang, M., Wang, T., Liu, Z., Tong, R., and Manocha, D. (2018). I-cloth: Incremental collision handling for gpu-based interactive cloth simulation. *ACM Transactions on Graphics (TOG)*, 37(6):1–10. 136

Tang, S., Chen, J., Wang, D., Tang, C., Zhang, F., Fan, Y., Chandra, V., Furukawa, Y., and Ranjan, R. (2025). Mvdiffusion++: A dense high-resolution multi-view diffusion model for single or sparse-view 3d object reconstruction. In *European Conference on Computer Vision*, pages 175–191. Springer. 249

Teran, J., Sifakis, E., Irving, G., and Fedkiw, R. (2005). Robust quasistatic finite elements and flesh simulation. In *Proceedings of the 2005 Symp. Computer animation*, pages 181–190. 21, 39, 93, 94, 110, 113, 286, 300, 301, 320

Terzopoulos, D. and Fleischer, K. (1988). Modeling inelastic deformation: viscolelasticity, plasticity, fracture. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 269–278. 15

Terzopoulos, D., Platt, J., Barr, A., and Fleischer, K. (1987). Elastically deformable models. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 205–214. 112

Tewari, A., Thies, J., Mildenhall, B., Srinivasan, P., Tretschk, E., Yifan, W., Lassner, C., Sitzmann, V., Martin-Brualla, R., Lombardi, S., et al. (2022). Advances in neural rendering. In *Computer Graphics Forum*, volume 41, pages 703–735. Wiley Online Library. 187

Thomaszewski, B., Pabst, S., and Strasser, W. (2009). Continuum-based strain limiting. In *Computer Graphics Forum*, volume 28, pages 569–576. Wiley Online Library. 138

Tjung, E. Y., Kularathna, S., Kumar, K., and Soga, K. (2020). Modeling irregular boundaries using isoparametric elements in material point method. In *Geo-Congress 2020: Modeling, Geomaterials, and Site Characterization*, pages 39–48. American Society of Civil Engineers Reston, VA. 77

Ton-That, Q.-M., Kry, P. G., and Andrews, S. (2023). Parallel block neo-hookean xpbd using graph clustering. *Computers & Graphics*, 110:1–10. 159

Tonge, R., Benevolenski, F., and Voroshilov, A. (2012). Mass splitting for jitter-free parallel rigid body simulation. *ACM Transactions on Graphics (TOG)*, 31(4):1–8. 139

Tournier, M., Nesme, M., Gilles, B., and Faure, F. (2015). Stable constrained dynamics. *ACM Trans. Graph.*, 34(4). 162

Tretschk, E., Tewari, A., Golyanik, V., Zollhöfer, M., Lassner, C., and Theobalt, C. (2021). Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *IEEE International Conference on Computer Vision (ICCV)*. IEEE. 188, 189, 201

Treuille, A., Lewis, A., and Popović, Z. (2006). Model reduction for real-time fluids. *ACM Transactions on Graphics (TOG)*, 25(3):826–834. 140

Trusty, T., Kaufman, D., and Levin, D. (2022). Mixed variational finite elements for implicit, general-purpose simulation of deformables. *arXiv:2202.00183*. 113

Turki, H., Agrawal, V., Bulò, S. R., Porzi, L., Kontschieder, P., Ramanan, D., Zollhöfer, M., and Richardt, C. (2023). Hybridnerf: Efficient neural rendering via adaptive volumetric surfaces. *arXiv preprint arXiv:2312.03160*. 228

Umetani, N., Schmidt, R., and Stam, J. (2015). Position-based elastic rods. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '14, page 21–30, Goslar, DEU. Eurographics Association. 159, 162

van Dijk, N. P., Maute, K., Langelaar, M., and Van Keulen, F. (2013). Level-set methods for structural topology optimization: a review. *Structural and Multidisciplinary Optimization*, 48(3):437–472. 282

Vlassis, N. N., Ma, R., and Sun, W. (2020). Geometric deep learning for computational mechanics part i: Anisotropic hyperelasticity. *Computer Methods in Applied Mechanics and Engineering*, 371:113299. 56

Vlassis, N. N. and Sun, W. (2021). Sobolev training of thermodynamic-informed neural networks for interpretable elasto-plasticity models with level set hardening. *Computer Methods in Applied Mechanics and Engineering*, 377:113695. 56

Volino, P., Courchesne, M., and Magnenat Thalmann, N. (1995). Versatile and efficient techniques for simulating cloth and other deformable objects. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 137–144. 139

Volino, P., Magnenat-Thalmann, N., and Faure, F. (2009). A simple approach to nonlinear tensile stiffness for accurate cloth simulation. *ACM Transactions on Graphics*, 28(4):Article–No. 139

Wächter, A. and Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57. 90, 94

Wallstedt, P. (2009). *On the order of accuracy of the generalized interpolation material point method*. 285

Wang, B., Wu, L., Yin, K., Ascher, U. M., Liu, L., and Huang, H. (2015). Deformation capture and modeling of soft objects. *ACM Trans. Graph.*, 34(4):94–1. 190

Wang, C., Jiang, R., Chai, M., He, M., Chen, D., and Liao, J. (2023a). Nerf-art: Text-driven neural radiance fields stylization. *IEEE Transactions on Visualization and Computer Graphics*. 227

Wang, H. (2015). A chebyshev semi-iterative approach for accelerating projective and position-based dynamics. *ACM Transactions on Graphics (TOG)*, 34(6):1–9. xiv, 138, 142, 148, 151

Wang, H., O'Brien, J., and Ramamoorthi, R. (2010). Multi-resolution isotropic strain limiting. *ACM Transactions on Graphics (TOG)*, 29(6):1–10. 138

Wang, H., O'Brien, J. F., and Ramamoorthi, R. (2011). Data-driven elastic models for cloth: modeling and measurement. *ACM transactions on graphics (TOG)*, 30(4):1–12. 139

Wang, H. and Yang, Y. (2016). Descent methods for elastic body simulation on the gpu. *ACM Trans. on Graph.*, 35(6):1–10. 21, 137

Wang, M. Y., Wang, X., and Guo, D. (2003). A level set method for structural topology optimization. *Computer methods in applied mechanics and engineering*, 192(1-2):227–246. 282, 285

Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., and Wang, W. (2021). Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*. 220, 226

Wang, P. and Shi, Y. (2023). Imagedream: Image-prompt multi-view diffusion for 3d generation. *arXiv preprint arXiv:2312.02201*. 249

Wang, T., Chen, J., Li, D., Liu, X., Wang, H., and Zhou, K. (2023b). Fast gpu-based two-way continuous collision handling. 140

Wang, W., Ho, H.-I., Guo, C., Rong, B., Grigorev, A., Song, J., Zarate, J. J., and Hilliges, O. (2024). 4d-dress: A 4d dataset of real-world human clothing with semantic annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 550–560. 271

Wang, X., Li, M., Fang, Y., Zhang, X., Gao, M., Tang, M., Kaufman, D. M., and Jiang, C. (2019). Hierarchical optimization time integration for cfl-rate mpm stepping. *arXiv preprint arXiv:1911.07913*. 286

Wang, X., Li, M., Fang, Y., Zhang, X., Gao, M., Tang, M., Kaufman, D. M., and Jiang, C. (2020a). Hierarchical optimization time integration for cfl-rate mpm stepping. *ACM Trans. on Graph.*, 39(3):1–16. 17, 21, 25, 39, 41, 56, 76, 83, 114, 160, 163, 328

Wang, X., Qiu, Y., Slattery, S., Fang, Y., Li, M., Zhu, S., Zhu, Y., Tang, M., Manocha, D., and Jiang, C. (2020b). A massively parallel and scalable multi-gpu material point method. *ACM Transactions on Graphics (TOG)*, 39(4):30–1. 114, 209

Wanger, L., Ferwerda, J., and Greenberg, D. (1992). Perceiving spatial relationships in computer-generated images. volume 12, pages 44–58. 235

Wardetzky, M., Bergou, M., Harmon, D., Zorin, D., and Grinspun, E. (2007). Discrete quadratic curvature energies. *Computer Aided Geometric Design*, 24(8-9):499–518. 139

Wathen, A. (2008). Chebyshev semi- iteration in preconditioning. Technical report. 138

Wendland, H. (1995). Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in computational Mathematics*, 4:389–396. 167

Westhofen, L., Jeske, S., and Bender, J. (2023). A comparison of linear consistent correction methods for first-order sph derivatives. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 6(3):1–20. 168

Wiewel, S., Becher, M., and Thuerey, N. (2019). Latent space physics: Towards learning the temporal evolution of fluid flow. In *Computer graphics forum*, volume 38, pages 71–82. Wiley Online Library. 141

Wojtan, C., Thürey, N., Gross, M., and Turk, G. (2009). Deforming meshes that split and merge. In *ACM SIGGRAPH 2009 papers*. 18

Wojtan, C. and Turk, G. (2008). Fast viscoelastic behavior with thin features. In *ACM SIGGRAPH 2008 papers*, pages 1–8. 18, 113

Wolfe, P. (1969). Convergence conditions for ascent methods. *SIAM review*, 11(2):226–235. 137

Wolfe, P. (1971). Convergence conditions for ascent methods. ii: Some corrections. *SIAM review*, 13(2):185–188. 137

Wolper, J., Chen, Y., Li, M., Fang, Y., Qu, Z., Lu, J., Cheng, M., and Jiang, C. (2020). Anisompm: Animating anisotropic damage mechanics: Supplemental document. *ACM Trans. Graph*, 39(4). 56, 113, 163

Wolper, J., Fang, Y., Li, M., Lu, J., Gao, M., and Jiang, C. (2019). Cd-mpm: continuum damage material point methods for dynamic fracture animation. *ACM Transactions on Graphics (TOG)*, 38(4):1–15. 54, 56, 110, 113, 160, 163, 174, 184, 285, 356, 357

Wolper, J., Gao, M., Lüthi, M. P., Heller, V., Vieli, A., Jiang, C., and Gaume, J. (2021). A glacier–ocean interaction model for tsunami genesis due to iceberg calving. *Communications Earth & Environment*, 2(1):1–10. 54, 55

Wu, B., Chen, Z., Zhang, X., Liu, Y., and Lian, Y. (2018). Coupled shell-material point method for bird strike simulation. *Acta Mechanica Solida Sinica*, 31(1):1–18. 75

Wu, B., Wang, Z., and Wang, H. (2022). A gpu-based multilevel additive schwarz preconditioner for cloth and deformable body simulation. *ACM Transactions on Graphics (TOG)*, 41(4):1–14. 113

Wu, G., Yi, T., Fang, J., Xie, L., Zhang, X., Wei, W., Liu, W., Tian, Q., and Wang, X. (2023a). 4d gaussian splatting for real-time dynamic scene rendering. *arXiv preprint arXiv:2310.08528*. 208, 210, 214, 221

Wu, J., Dick, C., and Westermann, R. (2015). A system for high-resolution topology optimization. *IEEE transactions on visualization and computer graphics*, 22(3):1195–1208. 282, 287

Wu, L., Wu, B., Yang, Y., and Wang, H. (2020). A safe and fast repulsion method for gpu-based cloth self collisions. *ACM Transactions on Graphics (TOG)*, 40(1):1–18. 136

Wu, T., Sun, J.-M., Lai, Y.-K., and Gao, L. (2023b). De-nerf: Decoupled neural radiance fields for view-consistent appearance editing and high-frequency environmental relighting. In *ACM SIGGRAPH 2023 conference proceedings*, pages 1–11. 227

Xia, H., Lin, Z.-H., Ma, W.-C., and Wang, S. Video2game: Real-time, interactive, realistic and browser-compatible environment from a single video. In *Synthetic Data for Computer Vision Workshop@ CVPR 2024*. 227

Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J. M., and Luo, P. (2021). Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in neural information processing systems*, 34:12077–12090. 260, 262

Xie, T., Li, M., Yang, Y., and Jiang, C. (2023a). A contact proxy splitting method for lagrangian solid-fluid coupling. *ACM Transactions on Graphics (TOG)*. 138, 140, 147

Xie, T., Zong, Z., Qiu, Y., Li, X., Feng, Y., Yang, Y., and Jiang, C. (2023b). Physgaussian: Physics-integrated 3d gaussians for generative dynamics. *arXiv preprint arXiv:2311.12198*. 225, 228, 230, 233, 234, 235, 239

Xie, Y., Takikawa, T., Saito, S., Litany, O., Yan, S., Khan, N., Tombari, F., Tompkin, J., Sitzmann, V., and Sridhar, S. (2022a). Neural fields in visual computing and beyond. *Computer Graphics Forum*. 187, 189

Xie, Y., Takikawa, T., Saito, S., Litany, O., Yan, S., Khan, N., Tombari, F., Tompkin, J., Sitzmann, V., and Sridhar, S. (2022b). Neural fields in visual computing and beyond. In *Computer Graphics Forum*, volume 41, pages 641–676. Wiley Online Library. 227

Xing, J., Ruan, L., Wang, B., Zhu, B., and Chen, B. (2022). Position-based surface tension flow. *ACM Trans. Graph.*, 41(6). 162

Xiu, Y., Yang, J., Cao, X., Tzionas, D., and Black, M. J. (2023). Econ: Explicit clothed humans optimized via normal integration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 512–523. 250

Xiu, Y., Yang, J., Tzionas, D., and Black, M. J. (2022). Icon: Implicit clothed humans obtained from normals. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13286–13296. IEEE. 250

Xu, H., Li, Y., Chen, Y., and Barbič, J. (2015). Interactive material design using model reduction. *ACM Transactions on Graphics (TOG)*, 34(2):1–14. 140

Xu, H., Zhao, Y., and Barbič, J. (2014). Implicit multibody penalty-baseddistributed contact. *IEEE transactions on visualization and computer graphics*, 20(9):1266–1279. 139

Xu, J., Chen, T., Zlokapa, L., Foshey, M., Matusik, W., Sueda, S., and Agrawal, P. (2021). An end-to-end differentiable framework for contact-aware robot design. *arXiv preprint arXiv:2107.07501*. 251

Xu, J., Du, T., Foshey, M., Li, B., Zhu, B., Schulz, A., and Matusik, W. (2019a). Learning to fly: computational controller design for hybrid uavs with reinforcement learning. *ACM Transactions on Graphics (TOG)*, 38(4):1–12. 318

Xu, J., Kim, S., Chen, T., Garcia, A. R., Agrawal, P., Matusik, W., and Sueda, S. (2023a). Efficient tactile simulation with differentiability for robotic manipulation. In *Conference on Robot Learning*, pages 1488–1498. PMLR. 251

Xu, L., Agrawal, V., Laney, W., Garcia, T., Bansal, A., Kim, C., Rota Bulò, S., Porzi, L., Kontschieder, P., Božič, A., et al. (2023b). Vr-nerf: High-fidelity virtualized walkable spaces. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–12. 228

Xu, Q., Xu, Z., Philip, J., Bi, S., Shu, Z., Sunkavalli, K., and Neumann, U. (2022). Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5438–5448. 207, 208

Xu, T. and Harada, T. (2022). Deforming radiance fields with cages. In *European Conference on Computer Vision*, pages 159–175. Springer. 205, 219, 221

Xu, Y., Chen, B., Li, Z., Zhang, H., Wang, L., Zheng, Z., and Liu, Y. (2023c). Gaussian head avatar: Ultra high-fidelity head avatar via dynamic gaussians. *arXiv preprint arXiv:2312.03029*. 225

Xu, Y., Tan, H., Luan, F., Bi, S., Wang, P., Li, J., Shi, Z., Sunkavalli, K., Wetzstein, G., Xu, Z., et al. (2023d). Dmv3d: Denoising multi-view diffusion using 3d large reconstruction model. *arXiv preprint arXiv:2311.09217*. 249

Xu, Z., Wu, J., Zeng, A., Tenenbaum, J. B., and Song, S. (2019b). Densephysnet: Learning dense physical object representations via multi-step dynamic interactions. *arXiv preprint arXiv:1906.03853*. 190, 201

Yang, J., Cheng, Z., Duan, Y., Ji, P., and Li, H. (2024). Consistnet: Enforcing 3d consistency for multi-view images diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7079–7088. 249

Yang, P., Gan, Y., Zhang, X., Chen, Z., Qi, W., and Liu, P. (2014). Improved decohesion modeling with the material point method for simulating crack evolution. *International Journal of Fracture*, 186(1-2):177–184. 285

Yang, S., He, X., and Zhu, B. (2020). Learning physical constraints with neural projections. *Advances in Neural Information Processing Systems*, 33:5178–5189. 54

Yang, T., Chang, J., Lin, M. C., Martin, R. R., Zhang, J. J., and Hu, S.-M. (2017a). A unified particle system framework for multi-phase, multi-material visual simulations. *ACM Trans. on Graph.*, 36(6):1–13. 20, 164

Yang, Y., Li, D., Xu, W., Tian, Y., and Zheng, C. (2015). Expediting precomputation for reduced deformable simulation. *ACM Trans. Graph. (TOG)*, 34(6). 140

Yang, Y., Sun, P., and Chen, Z. (2017b). Combined mpm-dem for simulating the interaction between solid elements and fluid particles. *Communications in Computational Physics*, 21(5):1258–1281. 74

Yang, Z., Gao, X., Zhou, W., Jiao, S., Zhang, Y., and Jin, X. (2023a). Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. *arXiv preprint arXiv:2309.13101*. 208

Yang, Z., Yang, H., Pan, Z., Zhu, X., and Zhang, L. (2023b). Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. *arXiv preprint arXiv:2310.10642*. 208, 225, 228

Yang, Z., Zeng, A., Yuan, C., and Li, Y. (2023c). Effective whole-body pose estimation with two-stages distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4210–4220. 259

Yariv, L., Gu, J., Kasten, Y., and Lipman, Y. (2021). Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems*, 34:4805–4815. 227

Ye, M., Danelljan, M., Yu, F., and Ke, L. (2023). Gaussian grouping: Segment and edit anything in 3d scenes. *arXiv preprint arXiv:2312.00732*. 223, 227

Yerro, A., Soga, K., and Bray, J. (2019). Runout evaluation of oso landslide with the material point method. *Canadian Geotechnical Journal*, 56(9):1304–1317. 285

Youtube (2018). Crushing long steel pipes with hydraulic press. 50

Youtube (2021). Satisfying steel pipe crush video. 50

Yu, Z., Dou, Z., Long, X., Lin, C., Li, Z., Liu, Y., Müller, N., Komura, T., Habermann, M., Theobalt, C., et al. (2023). Surf-d: High-quality surface generation for arbitrary topologies using diffusion models. *arXiv preprint arXiv:2311.17050*. 249

Yuan, Y.-J., Sun, Y.-T., Lai, Y.-K., Ma, Y., Jia, R., and Gao, L. (2022). Nerf-editing: geometry editing of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18353–18364. 205, 208, 219, 221, 222

Yue, Y., Smith, B., Batty, C., Zheng, C., and Grinspun, E. (2015). Continuum foam: A material point method for shear-dependent flows. *ACM Trans. on Graph.*, 34(5):160. 18, 113, 163, 174, 190, 218, 358

Yue, Y., Smith, B., Chen, P. Y., Chantharayukhonthorn, M., Kamrin, K., and Grinspun, E. (2018). Hybrid grains: Adaptive coupling of discrete and continuum simulations of granular media. In *SIGGRAPH Asia 2018 Technical Papers*, page 283. ACM. 19, 176

Zabala, F. and Alonso, E. (2011). Progressive failure of aznalcóllar dam using the material point method. *Géotechnique*, 61(9):795–808. 285

Zhang, C., Wang, Y., Vicente, F., Wu, C., Yang, J., Beeler, T., and De la Torre, F. (2024). Fabricdiffusion: High-fidelity texture transfer for 3d garments generation from in-the-wild images. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–12. 270

Zhang, D. Z. (2013). Dual domain material point method for extreme material deformation. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States). 85

Zhang, D. Z., Ma, X., and Giguere, P. T. (2011). Material point method enhanced by modified gradient of shape function. *Journal of Computational Physics*, 230(16):6379–6398. 73

Zhang, D. Z., Zou, Q., VanderHeyden, W. B., and Ma, X. (2008). Material point method applied to multiphase flows. *Journal of Computational Physics*, 227(6):3159–3173. 73

Zhang, F., Zhang, X., Sze, K. Y., Lian, Y., and Liu, Y. (2017a). Incompressible material point method for free surface flow. *Journal of Computational Physics*, 330:92–110. 73, 285

Zhang, H., Wang, K., and Chen, Z. (2009). Material point method for dynamic analysis of saturated porous media under external contact/impact of solid bodies. *Computer methods in applied mechanics and engineering*, 198(17-20):1456–1472. 285

Zhang, L., Rao, A., and Agrawala, M. (2023). Adding conditional control to text-to-image diffusion models. 249

Zhang, W., Chen, J., Zhu, X., Zhou, J., Xue, D., Lei, X., and Guo, X. (2017b). Explicit three dimensional topology optimization via moving morphable void (mmv) approach. *Computer Methods in Applied Mechanics and Engineering*, 322:590–614. 283

Zhang, W., Li, D., Yuan, J., Song, J., and Guo, X. (2017c). A new three-dimensional topology optimization method based on moving morphable components (mmcs). *Computational Mechanics*, 59(4):647–665. 283

Zhang, W., Li, D., Zhang, J., and Guo, X. (2016a). Minimum length scale control in structural topology optimization based on the moving morphable components (MMC) approach. *Computer Methods in Applied Mechanics and Engineering*, 311:327–355. 283

Zhang, W., Song, J., Zhou, J., Du, Z., Zhu, Y., Sun, Z., and Guo, X. (2017d). Topology optimization with multiple materials via moving morphable component (MMC) method. *International Journal for Numerical Methods in Engineering*, 113(11):1653–1675. 283

Zhang, W., Zhang, J., and Guo, X. (2016b). Lagrangian description based topology optimization—a revival of shape optimization. *Journal of Applied Mechanics*, 83(4):041010. 283

Zhang, X., Chen, Z., and Liu, Y. (2016c). *The material point method: a continuum-based particle method for extreme loading cases*. Academic Press. 73, 79, 283

Zhang, X., Sze, K., and Ma, S. (2006). An explicit material point finite element method for hyper-velocity impact. *International Journal for Numerical Methods in Engineering*, 66(4):689–706. 74

Zhao, F., Wang, W., Liao, S., and Shao, L. (2021). Learning anchored unsigned distance functions with gradient direction alignment for single-view garment reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12674–12683. 250

Zhao, M., Anzai, T., Shi, F., Chen, X., Okada, K., and Inaba, M. (2018a). Design, modeling, and control of an aerial robot dragon: A dual-rotor-embedded multilink robot with the ability of multi-degree-of-freedom aerial transformation. *IEEE Robotics and Automation Letters*, 3(2):1176–1183. 318

Zhao, M., Kawasaki, K., Anzai, T., Chen, X., Noda, S., Shi, F., Okada, K., and Inaba, M. (2018b). Transformable multirotor with two-dimensional multilinks: Modeling, control, and whole-body aerial manipulation. *The International Journal of Robotics Research*, 37(9):1085–1112. 318

Zhao, Y., Choo, J., Jiang, Y., and Li, L. (2023a). Coupled material point and level set methods for simulating soils interacting with rigid objects with complex geometry. *Computers and Geotechnics*, 163:105708. 129

Zhao, Y., Choo, J., Jiang, Y., Li, M., Jiang, C., and Soga, K. (2022). A barrier method for frictional contact on embedded interfaces. *Computer Methods in Applied Mechanics and Engineering*, 393:114820. 57, 76

Zhao, Y., Jiang, C., and Choo, J. (2023b). Circumventing volumetric locking in explicit material point methods: A simple, efficient, and general approach. *International Journal for Numerical Methods in Engineering*, 124(23):5334–5355. 133

Zheng, Y., Zhao, Q., Yang, G., Yifan, W., Xiang, D., Dubost, F., Lagun, D., Beeler, T., Tombari, F., Guibas, L., et al. (2025). Physavatar: Learning the physics of dressed 3d avatars from visual observations. In *European Conference on Computer Vision*, pages 262–284. Springer. 252

Zhu, H., Cao, Y., Jin, H., Chen, W., Du, D., Wang, Z., Cui, S., and Han, X. (2020). Deep fashion3d: A dataset and benchmark for 3d garment reconstruction from single images. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 512–530. Springer. 250

Zhu, Y. and Bridson, R. (2005). Animating sand as a fluid. *ACM Transactions on Graphics (TOG)*, 24(3):965–972. 173, 226

Zielonka, W., Bagautdinov, T., Saito, S., Zollhöfer, M., Thies, J., and Romero, J. (2023). Drivable 3d gaussian avatars. *arXiv preprint arXiv:2311.08581*. 225

Zimmerman, B. K. and Ateshian, G. A. (2018). A surface-to-surface finite element algorithm for large deformation frictional contact in febio. *Journal of Biomechanical Engineering*, 140(8). 105, 106

Zong, Z., Li, X., Li, M., Chiaramonte, M. M., Matusik, W., Grinspun, E., Carlberg, K., Jiang, C., and Chen, P. Y. (2023). Neural stress fields for reduced-order elastoplasticity and fracture. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–11. 217