
Gaussian-Augmented Physics Simulation and System Identification with Complex Colliders

Federico Vasile¹ Ri-Zhao Qiu² Lorenzo Natale¹ Xiaolong Wang²

¹Istituto Italiano di Tecnologia, ²UC San Diego

<https://as-diffmpm.github.io>



Figure 1: We present the **Any-Shape Differentiable Material Point Method (AS-DiffMPM)**, a particle-based framework for simulating collisions with **arbitrarily shaped** rigid bodies. When integrated with Gaussian-based rendering, it enables physically plausible **animation** and **system identification** (estimation of object physical parameters from visual observations).

Abstract

System identification involving the geometry, appearance, and physical properties from video observations is a challenging task with applications in robotics and graphics. Recent approaches have relied on fully differentiable Material Point Method (MPM) and rendering for simultaneous optimization of these properties. However, they are limited to simplified object-environment interactions with planar colliders and fail in more challenging scenarios where objects collide with non-planar surfaces. We propose AS-DiffMPM, a differentiable MPM framework that enables physical property estimation with arbitrarily shaped colliders. Our approach extends existing methods by incorporating a differentiable collision handling mechanism, allowing the target object to interact with complex rigid bodies while maintaining end-to-end optimization. We show AS-DiffMPM can be easily interfaced with various novel view synthesis methods as a framework for system identification from visual observations.

1 Introduction

Bending but not breaking, a branch dances with the wind—yielding to its force yet never surrendering. From the gentle drift of a balloon to the violent shatter of glass, the world reveals its hidden truths through motion. We, too, learn not only by *seeing* but by *understanding*—tracing back from movement to the forces that shaped it. In this work, we give machines this same power: *to see, reason, and uncover the physical essence* of objects through their dance with the world, recovering their hidden properties from mere pixels.

Estimating the physical parameters of objects from visual observations is a challenging task in computer vision, with applications in robotics, virtual reality, and graphics. While existing methods

such as PhysGaussian [1] can **animate** static scenes by simulating plausible dynamics, they do not support the **identification** of object properties from dynamic scenes. Humans naturally infer an object’s geometry and dynamics through a two-step process. First, they recognize and understand its shape (e.g., a ball). Then, by observing its interactions with the environment (e.g., let the ball fall on the ground), they deduce its physical properties. Inspired by this intuition, modern system identification methods follow a similar paradigm, combining two components: a *rendering model* to reconstruct geometry from images and a *physics engine* to simulate motion, enabling the estimation of physical properties from video sequences.

Recent advances in scene representations — particularly point-based models [2]—and particle-based physics simulation [3], have facilitated the animation of static geometry with physics-grounded motion. This process involves establishing correspondences between the points used for rendering and the particles in simulation, allowing the radiance field to be animated by applying the motion derived from simulated particle trajectories. As a result, for physical properties estimation, this approach can be applied by reconstructing the object’s geometry, initializing an estimate of its physical parameters, and refining them through gradient-based optimization, comparing simulated motion against real-world observations.

PAC-NeRF [4] is a pioneering model for system identification from visual observations; combining voxel-based radiance fields [5] with a Differentiable Material Point Method (DiffMPM) [6] to enable gradient-based optimization of physical parameters from images. However, this approach, along with other recent works [7, 8, 9], share a significant limitation: they perform system identification only when the object interacts with simple boundaries such as the ground. While effective, this setup fails to capture more complex scenarios, where objects interact with arbitrarily shaped rigid bodies.

In this work, we extend system identification beyond simple boundary interactions, enabling estimation of physical properties in more complex settings. To the best of our knowledge, this problem has not been previously addressed, primarily due to two key limitations: (i) existing system identification methods are restricted to interactions with a planar boundary [4, 7, 8, 9], and (ii) available MPM-based simulators that accurately handle arbitrarily shaped colliders are not differentiable [10], thus not suitable for physical parameters optimization. Our contribution is threefold:

1. **Any-Shape Differentiable MPM.** We propose AS-DiffMPM, a Differentiable MPM capable of handling collisions between the target object and arbitrarily shaped colliders. Quantitative experiments on *Newtonian*, *non-Newtonian* and *granular* materials demonstrate the necessity of an ad hoc solution for rigid body interactions, which previous system identification methods have overlooked.
2. **Versatile Collision Handling Framework.** We design a general interface to integrate colliders of various formats (i.e., mesh and 2D Gaussians [11]) into AS-DiffMPM.
3. **Rendering-Physics Integration for System Identification.** We combine our AS-DiffMPM with multiple rendering models and establish a benchmark for system identification from visual observations when the object collides with arbitrarily shaped rigid bodies.

2 Related Work

Scene Representations for Photorealistic Rendering. The topic of scene reconstruction and synthesizing photorealistic novel views has been studied in the computer vision community for decades [12, 13, 14]. Recently, there has been a growing interest in using learning-based differentiable rendering or rasterization for such a purpose [2, 15, 16, 17, 18]. These methods are often categorized into implicit rendering methods based on ray-marching queries [15, 16, 17, 18] or explicit rasterization methods [2, 11]. To manipulate and animate these static scene representations, researchers have shown that it is possible to edit both implicit methods (often via remapping voxel-based representations [4, 19]) and explicit methods (via direct manipulation of explicit units [8, 1, 20]). Our method is mostly orthogonal to the development in scene representations. The AS-DiffMPM is a plug-and-play module to both NeRF- and 3DGS-based methods, as quantitatively shown in the experiment.

Physics-integrated Scene Synthesis. While high-quality reconstruction methods offer a pathway to synthesize *static* scenes from a set of images in novel viewpoints, they are limited to understanding static appearances. Humans, on the other hand, have the ability to picture physically plausible interactions with the scenes. To equip machines with such an intelligence, researchers have sought to animate static scenes with physics simulators to generate animated scenes that are both visually and

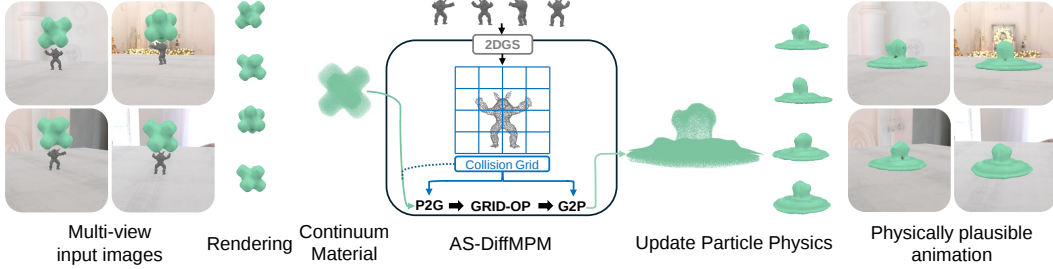


Figure 2: **Overview.** Given multi-view input images, we separately reconstruct the continuum object using a rendering model (e.g., [5, 11]) and the rigid body collider with 2DGS [11]. Particle trajectories are subsequently advected using AS-DiffMPM. Finally, the updated particle positions are mapped back to rendering primitives for photorealistic and physically plausible animation.

physically appealing [1, 21, 20, 22, 4, 23, 24]. Due to the indirection caused by implicitness, few methods animate implicit NeRFs [23]. Many researchers have combined Material Point Method [3] with Gaussian Splatting [2] due to their naturally aligned point-based representations. These methods have shown to provide photorealistic animations [1, 25] with extensions to VR applications [22], language-driven animations [20], and complex fluid dynamics [21]. However, these methods provide physically plausible **animations of static scenes** rather than estimating physical parameters of objects. Another line of work leverages object dynamics priors from video generative models to endow **static 3D objects with interactive behaviors** [26, 27, 28, 29]. While effective for visually realistic interactions—useful in applications such as content creation—they may fail in domains like robotics, where precise physical understanding is necessary. In contrast, our work aims to recover accurate physical parameters of objects from visual observations.

System Identification from Videos. The identification of dynamic systems goes beyond simple animation, as it requires not only visually plausible animations but also accurate physical properties that replicate the behavior of the reference system. Prior to the introduction of photorealistic differentiable rendering [15], system identifications from videos have been difficult due to the coupling of appearance and dynamics [30]. Recent works have leveraged gradient-based optimization of physical parameters through differentiable rendering and simulation for system identification. PAC-NeRF [4] is a representative method that combines a voxel-based NeRF [5] and DiffMPM for this task. Following up to this work, GIC [8] introduces a novel scene representation method upon 3DGS [2] to improve the supervisory signal from visual observations. Additionally, [9] replaces MPM with a Spring-Mass representation to model elastic objects. However, these methods focus on system identification under the assumption of object interactions with planar surfaces, as the underlying physics simulator does not support complex colliders. Our AS-DiffMPM bridges this gap by supporting system identification under more realistic conditions, such as collisions with complex colliders.

3 Preliminary

3.1 Material Point Method

The Material Point Method (MPM) [31, 32, 3] is a widely adopted approach to simulate the dynamics of continuum materials. It combines the strengths of both Lagrangian particles and Eulerian grid by discretizing the continuum as a set of particles carrying physical information such as position, velocity, and mass. For every update step, these properties are transferred to the Eulerian background grid to solve the equations of motion on the grid nodes. Finally, the resulting nodal velocities on the grid nodes are interpolated back to the Lagrangian particles, thus updating the state of the material. MPM operates in three stages: *Particle-to-Grid (P2G)* transfer, *Grid Operations (G-OP)*, and *Grid-to-Particle (G2P)* transfer.

Specifically, considering a toy example where an object is subject to gravity, we represent the object as a set of particles carrying the following Lagrangian quantities: the particle position $\mathbf{x}_p$, velocity $\mathbf{v}_p$, and mass m_p . These particles interact with an Eulerian grid, which stores nodal quantities: the position $\mathbf{x}_g$, mass m_g and velocity $\mathbf{v}_g$ at the grid node g .

At each time step, the simulation proceeds through the following stages:

- *Particle-to-Grid (P2G)*: Physical quantities from particles are transferred to nearby grid nodes using a weighting function $w_g(\mathbf{x}_g - \mathbf{x}_p)$. The mass and momentum of each particle contribute to the grid:

$$m_g = \sum_p m_p w_g(\mathbf{x}_g - \mathbf{x}_p), \quad \mathbf{v}_g = \frac{1}{m_g} \sum_p m_p \mathbf{v}_p w_g(\mathbf{x}_g - \mathbf{x}_p). \quad (1)$$

- *Grid Operations (G-OP)*: Once the mass and momentum fields are constructed on the grid, forces such as gravity $\mathbf{g}$ are applied to update velocities at the grid nodes: $\mathbf{v}_g \leftarrow \mathbf{v}_g + \Delta t \mathbf{g}$. Next, boundary conditions are enforced to ensure that the object correctly interacts with obstacles in the scene. In this example, we consider a *sticky ground*, meaning any particle colliding with the surface loses velocity upon impact. The ground is represented as a plane defined by a point $\mathbf{x}_b$ and a normal $\mathbf{n}_b$. A grid node g is in contact with the ground if $(\mathbf{x}_g - \mathbf{x}_b) \cdot \mathbf{n}_b \leq 0$, in which case its velocity is set to zero, i.e., $\mathbf{v}_g = \mathbf{0}$.
- *Grid-to-Particle (G2P)*: The velocity field from the grid nodes is interpolated back to the particles to update their state. Each particle gathers velocity from the surrounding grid nodes:

$$\mathbf{v}_p = \sum_g \mathbf{v}_g w_g(\mathbf{x}_g - \mathbf{x}_p). \quad (2)$$

The particle positions are then updated: $\mathbf{x}_p \leftarrow \mathbf{x}_p + \Delta t \mathbf{v}_p$.

3.2 System identification

System identification aims to estimate the geometric structure and physical properties of dynamic objects from multi-view video sequences. Following prior works [4, 7, 8], we assume that the object material (e.g., Newtonian) is known and that its behavior adheres to continuum mechanics [32, 33].

4 Rigid Body Colliders

4.1 Collision Resolution in MPM

A common approach for handling collisions between the continuum material and a rigid body is to apply velocity corrections during the *G-OP* stage, as introduced in Sec. 3.1. Specifically, level sets (or signed distance functions) are used to classify each grid node as inside or outside the collider and adjust the nodal velocity accordingly [31]. Although straightforward and effective, this method fails with complex geometries (e.g., open surfaces, sharp boundaries), since all the particles in the same grid-cell region share a single velocity field. In contrast, Compatible Particle-in-Cell (CPIC) [10] handles collisions in a particle-wise manner during the *P2G* and *G2P* steps. It partitions grid nodes and particles into compatible or incompatible sets relative to the boundary’s surface, allowing more precise velocity corrections for each particle colliding with the rigid body.

Hereafter, we detail the CPIC [10] method to project the rigid body onto a grid-based distance field (Sec. 4.1.1). This information is transferred to the continuum material (Sec. 4.1.2) and used to possibly apply velocity corrections on particles based on their relationship with neighboring grid nodes (Sec. 4.1.3).

4.1.1 Project Rigid Body to Collision Grid

Following prior work [10], we represent the rigid body collider as a mesh and project it onto a grid that encodes properties for collision handling. We define a Collision Grid with the same resolution as the MPM Eulerian grid and store the following quantities at each grid node: (i) the *affinity* flag A_g indicating whether the grid node g is near the boundary, (ii) the *unsigned distance* d_g from the grid node to the boundary, (iii) a *tag* T_g denoting which side of the boundary the grid node is on and (iv) a *normal* $\mathbf{n}_g$ retrieved from the boundary. The projection process follows three key steps: sampling rigid particles, identifying affinity nodes and reconstructing the Collision Grid.

Sampling rigid particles. To map the rigid body’s surface onto nearby grid nodes, the collider is treated as a collection of primitives, thus, we sample a predefined number of rigid particles on each

face ξ^i of the mesh representing the rigid body. Moreover, we define $\xi(\mathbf{x}_{rp})$ as the face to which the rigid particle $\mathbf{x}_{rp}$ belongs.

Identifying affinity nodes. In order to identify all the grid nodes surrounding the rigid body, we map each $\mathbf{x}_{rp}$ to the $3 \times 3 \times 3$ grid of neighboring grid nodes denoted as $\mathcal{N}(\mathbf{x}_{rp})$. For each grid node $g \in \mathcal{N}(\mathbf{x}_{rp})$, its position $\mathbf{x}_g$ is projected onto the plane defined by the face $\xi(\mathbf{x}_{rp})$. If the projection falls inside the face, we store the $(\mathbf{x}_g, \xi(\mathbf{x}_{rp}))$ pair and set $A_g = 1$ (otherwise, we set $A_g = 0$).

Reconstructing the Collision Grid. Since there may be multiple faces in proximity to the same affinity grid node, i.e., $\{(\mathbf{x}_g, \xi^i)\}, i \in \{1, \dots, l\}$, we select the one with minimum unsigned point-plane distance:

$$\xi_g^* = \arg \min_{i \in \{1, \dots, l\}} |\text{dist}(\mathbf{x}_g, \xi^i)|, \quad (3)$$

and store its normal in $\mathbf{n}_g$, the sign of the distance in $T_g = \text{sign}(\text{dist}(\mathbf{x}_g, \xi_g^*))$ and the unsigned distance in d_g .

In summary, for each grid node g , the Collision Grid carries the following properties: $A_g, d_g, T_g, \mathbf{n}_g$.

4.1.2 Transfer Collision Grid to Material Particles

The Collision Grid is used to transfer properties to material particles, ensuring accurate interactions with the boundary and possibly correcting undesired behaviors (e.g., penetration).

Particle affinity. Given the material particle position $\mathbf{x}_p$, we set $A_p = 1$ if at least one of the $3 \times 3 \times 3$ neighboring grid nodes $g \in \mathcal{N}(\mathbf{x}_p)$ has $A_g = 1$.

Particle distance, tag, and normal. For each material particle $\mathbf{x}_p$ in affinity with the Collision Grid, we retrieve the collision properties through interpolation over $\mathcal{N}(\mathbf{x}_p)$:

$$d_p = \sum_{g \in \mathcal{N}(\mathbf{x}_p)} w_g(\mathbf{x}_g - \mathbf{x}_p) A_g T_g d_g, \quad \mathbf{n}_p = \sum_{g \in \mathcal{N}(\mathbf{x}_p)} w_g(\mathbf{x}_g - \mathbf{x}_p) A_g \mathbf{n}_g, \quad (4)$$

where w is a nodal weighting function (e.g., B-spline [31, 34]). Then, we set $T_p = \text{sign}(d_p)$. However, if a particle penetrates the boundary, an incorrect T_p may be reconstructed. Therefore, each particle's tag T_p retains its first acquired value until the particle loses affinity, i.e., moves away from the boundary. As a result, if a particle penetrates the boundary due to simulation errors, it will have an incorrect d_p (i.e., wrong sign) while still retaining a correct T_p . This allows penetrations to be fixed: $\mathbf{f}_p = -k_h d_p \mathbf{n}_p$, where $\mathbf{f}_p$ is the penalty force to apply to the material particle p and k_h is the penalty stiffness parameter.

Particle-grid compatibility. When a material particle approaches the boundary, its tag T_p is compared with the neighboring nodes $g \in \mathcal{N}(\mathbf{x}_p)$: p and g are incompatible if $T_p = 1$ and $T_g = -1$, or vice versa. In other words, they are on two different sides of the boundary. This condition is used during the *P2G* and *G2P* (Sec. 4.1.3) to identify the particles that are about to collide and adjust their velocities in a particle-wise manner, rather than applying nodal velocities during the *G-OP* stage.

4.1.3 P2G and G2P with CPIC

During the *P2G* stage, material particles transfer velocities only to compatible grid nodes. Consequently, no collision handling occurs during the *G-OP* stage, as nodal velocities receive contributions solely from particles away from the boundary. In the *G2P* stage, however, material particles gather velocities from both compatible and incompatible grid nodes. For each incompatible node, we directly reuse the particle's current velocity $\mathbf{v}_p$, enabling particle-wise collision handling with the surface. For instance, assuming a slippery surface, the velocity is projected onto the surface as $\mathbf{v}_p^{\text{proj}} = \mathbf{v}_p - (\mathbf{v}_p \cdot \mathbf{n}_p) \mathbf{n}_p$.

4.2 Representing Rigid Body as 2D Gaussians

As discussed in Sec. 4.1.1, previous work represents the rigid body collider as a mesh [10]. In contrast, we design a generalized interface for our framework, enabling seamless integration of both meshes and 2D Gaussians [11] as colliders. By utilizing the Collision Grid as a proxy for transferring collision properties from the rigid body to the material particles, our framework can accommodate any collider represented as a set of primitives ξ^i with associated normals $\mathbf{n}_i$.

We reconstruct the rigid body from multi-view images using 2D Gaussian Splatting [11]. Thus, in this case, the rigid body primitive ξ^i corresponds to a planar disk (i.e., 2D Gaussian) rather than a mesh face. The collider is imported into AS-DiffMPM following the procedure in Sec. 4.1, with minimal adaptation required for identifying the affinity grid nodes (Sec. 4.1.1).

Sampling rigid particles. We follow the previously described procedure to sample rigid particles $\mathbf{x}_{rp}$ on the primitives ξ^i .

Identifying affinity nodes. The $3 \times 3 \times 3$ neighboring grid nodes $\mathcal{N}(\mathbf{x}_{rp})$ of rigid particle $\mathbf{x}_{rp}$ are projected onto the plane defined by the planar disk ξ^i . If the projection falls inside the disk, we store the $(\mathbf{x}_g, \xi(\mathbf{x}_{rp}))$ and set the affinity flag A_g accordingly.

From this stage onward, the procedure adheres to the presented approach for transferring collision properties to material particles and resolving collisions during the *P2G* and *G2P* stages.

4.3 System Identification

We integrate our AS-DiffMPM with multiple rendering methods to enable system identification from visual observations.

Voxel-based NeRF. We follow [4] to integrate a voxel-based NeRF [5] with AS-DiffMPM. Both MPM and voxel-based NeRFs rely on a grid view (G) for computation and rendering, respectively. However, while MPM uses a Lagrangian view (P) to advect particles representing the continuum material, voxel-based NeRFs lack an equivalent point-based representation. Thus, a mapping between voxels and Lagrangian particles is required to enable dynamic rendering. This process involves: (i) mapping voxel fields $\mathcal{F}_i^G$ to Lagrangian particle fields $\mathcal{F}_p^P$, (ii) advecting particles using MPM and (iii) mapping the updated particle positions back to voxels for dynamic rendering. The following interconverters for G and P views are used:

$$\mathcal{F}_p^P \approx \sum_i w_{ip} \mathcal{F}_i^G, \quad \mathcal{F}_i^G \approx \frac{\sum_p w_{ip} \mathcal{F}_p^P}{\sum_p w_{ip}}, \quad (5)$$

where p and i indicate the index of the particle and grid node, respectively; and w_{ip} is a weighting function defined on i and evaluated at p . As a result, this mapping mechanism enable bidirectional transfer between the voxel grid and Lagrangian particles, allowing for gradient-based optimization of physical parameters from visual observations.

Point-based methods. Point-based rendering methods [8, 2, 11] naturally integrate with MPM as they both utilize the Lagrangian view (P) for rendering and particle advection. By employing differentiable point-based primitives [2, 11], the gradient of the photometric loss with respect to each rendering primitive is computed and propagated to its corresponding Lagrangian particle, thereby supporting gradient-based optimization of the physical parameters from visual observations. Formally:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \sum_r \sum_p \underbrace{\left(\frac{\partial \mathcal{L}}{\partial I} \cdot \frac{\partial I}{\partial \mathbf{x}_r} \right)}_{\text{Rendering}} \cdot \underbrace{\frac{\partial \mathbf{x}_r}{\partial \mathbf{x}_p^r}}_{\text{Mapping}} \cdot \underbrace{\frac{\partial \mathbf{x}_p^r}{\partial \theta}}_{\text{MPM}}, \quad (6)$$

where $\mathcal{L}$ is the photometric loss comparing the rendered image I with the ground truth, $\mathbf{x}_r$ denotes the location of the rendering primitive r , $\mathbf{x}_p^r$ indicates the position of the Lagrangian particle p associated with primitive r and θ represents the physical parameters (e.g., stiffness, viscosity). The final gradient is the product of three components: (i) rendering, (ii) primitive-particle mapping, and (iii) differentiable MPM. For simplicity, each Lagrangian particle is initialized from—and tied to—a rendering primitive (i.e., $\mathbf{x}_p^r = \mathbf{x}_r$), reducing the gradient flow to:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \sum_p \left(\frac{\partial \mathcal{L}}{\partial I} \cdot \frac{\partial I}{\partial \mathbf{x}_p} \right) \cdot \frac{\partial \mathbf{x}_p}{\partial \theta}. \quad (7)$$



Figure 3: Qualitative examples of reference frames used in experiments. Note that previous works [4, 7, 8, 9] perform system identification only with a planar surface and do not support such colliders.

5 Experiments

Physical Parameters. Our analysis focuses on material types that undergo noticeable deformation upon collision with rigid bodies, specifically Newtonian fluids, non-Newtonian fluids, and granular media. For Newtonian fluids, we estimate fluid viscosity (μ) and bulk modulus (κ). For non-Newtonian fluids, we recover shear modulus (μ), bulk modulus (κ), yield stress (τ_Y), and plasticity viscosity (η). For granular media, we estimate the friction angle (θ_{fric}). For a comprehensive overview of physical parameters and constitutive models, we refer to [4].

Dataset. We follow the protocol in [4] and generate ground-truth simulation rollouts using the cross-shaped object from [4] as our continuum material, with the physical parameter values provided in Appendix E. The object undergoes free fall and collides with a static rigid body (*Box*, *Bunny*, or *Armadillo*) having sticky surfaces. For each rollout, we collect both the 3D particle trajectories (i.e., point clouds over time) of the continuum material and the corresponding rendered frames (see Fig. 3), captured from 11 cameras placed around the scene. These data are used in Sec. 5.1 and Sec. 5.2 for experiments on system identification. Following the protocol in [4], we generate 10 rollouts per collider for each material type, except for granular media, where 5 videos are generated, resulting in a total of 75 rollouts. Each rollout is 16 timesteps long.

Training and Evaluation. Our training setup also adheres to [4], including the use of the Adam optimizer [35] and the initial guesses for the physical parameters. Unlike [4], we do not optimize the initial velocity vector during the free-fall phase; instead, we use the ground truth values, as our focus is on system identification during collisions. Finally, we report the results of physical parameters estimation using the mean and standard deviation of absolute errors, scaled by a factor of 100.

5.1 System Identification from Particle Trajectories

Training. This experiment evaluates system identification performance by estimating the physical parameters of the continuum material using the 3D particle trajectory as supervision. Specifically, we compute a particle-wise Mean Squared Error (MSE) loss between the reference trajectory and the one simulated by the MPM using the current parameter estimates. The loss is backpropagated through the differentiable simulator to iteratively optimize the physical parameters. We emphasize that this experiment does not involve visual observations or rendering models, as the focus is exclusively on evaluating the effectiveness of different collision handling strategies for system identification.

Baselines. As discussed in Sec. 4.1, a straightforward approach for collision handling in MPM is during the *G-OP* stage, where a Signed Distance Function (SDF) is reconstructed from the rigid body collider [27]. Assuming sticky surfaces, the nodal velocities for grid nodes inside the collider are set to zero. We refer to this baseline as GOP-DiffMPM. Additionally, following prior works [1, 36], the rigid collider can be represented as a set of rigid particles. We denote this approach as RP-DiffMPM.

Results. Tab. 1 compares the performance of AS-DiffMPM with baselines. Notably, no consistent trend is observed across colliders: even geometrically simpler shapes like the Box can present

Material	Collider	Parameters	RP-DiffMPM	GOP-DiffMPM	AS-DiffMPM (Ours)
Newtonian	Box	$\log_{10}(\mu)$	4.82 ± 5.15	2.73 ± 3.95	5.08 ± 2.73
		$\log_{10}(\kappa)$	37.97 ± 21.8	17.09 ± 19.9	27.81 ± 33.49
	Bunny	$\log_{10}(\mu)$	6.69 ± 5.97	0.43 ± 0.3	4.35 ± 5.24
		$\log_{10}(\kappa)$	35.04 ± 37.35	11.52 ± 15.97	31.37 ± 50.02
	Armadillo	$\log_{10}(\mu)$	5.67 ± 3.98	0.83 ± 1.14	0.4 ± 0.76
		$\log_{10}(\kappa)$	34.61 ± 46.52	13.08 ± 10.66	0.22 ± 0.24
Non-Newtonian	Box	$\log_{10}(\mu)$	42.01 ± 32.76	41.11 ± 27.78	23.51 ± 30.07
		$\log_{10}(\kappa)$	99.55 ± 98.64	136.04 ± 78.32	27.06 ± 19.12
		$\log_{10}(\tau_Y)$	20.13 ± 25.83	5.86 ± 3.21	7.28 ± 3.75
		$\log_{10}(\eta)$	52.38 ± 27.63	53.22 ± 42.22	17.7 ± 15.17
	Bunny	$\log_{10}(\mu)$	53.02 ± 47.05	20.29 ± 13.96	21.92 ± 36.89
		$\log_{10}(\kappa)$	82.83 ± 86.55	134.82 ± 75.71	26.77 ± 25.09
		$\log_{10}(\tau_Y)$	31.86 ± 50.01	7.10 ± 2.72	2.48 ± 1.87
		$\log_{10}(\eta)$	56.01 ± 31.13	53.36 ± 36.15	39.1 ± 36.2
	Armadillo	$\log_{10}(\mu)$	36.67 ± 20.92	36.81 ± 34.27	7.14 ± 6.64
		$\log_{10}(\kappa)$	192.43 ± 76.69	176.31 ± 73.89	56.26 ± 76.97
		$\log_{10}(\tau_Y)$	13.72 ± 4.58	12.10 ± 5.29	5.07 ± 5.79
		$\log_{10}(\eta)$	64.40 ± 40.39	54.47 ± 38.32	44.98 ± 41.34
Granular	Box	θ_{fric}	1.58 ± 0.65	1.16 ± 0.84	1.22 ± 0.71
	Bunny	θ_{fric}	1.91 ± 0.42	0.11 ± 0.15	0.06 ± 0.07
	Armadillo	θ_{fric}	2.28 ± 0.66	0.26 ± 0.16	0.51 ± 0.96

Table 1: **System identification from particle trajectories.** We compare AS-DiffMPM with two differentiable baselines for collision handling with arbitrarily shaped rigid bodies. Each method is evaluated based on its ability to recover physical parameters using particle trajectories as supervision. Best (light red) and second best (very light red) results are highlighted.

challenges due to sharp corners, while more complex shapes like the Bunny and Armadillo have intricate surfaces leading to complex particle trajectories. Overall, AS-DiffMPM achieves comparable or superior performance, due to its particle-wise collision handling. In contrast, GOP-DiffMPM may be less accurate with complex geometries, as it assigns a single velocity field to all particles within the same grid cell. Finally, RP-DiffMPM consistently underperforms, suggesting that accurate system identification requires tailored collision resolution strategies.

5.2 System Identification from Visual Observations

Training. We follow the system identification pipeline outlined in [4]. Given a multi-view video, we first segment the continuum object using video matting techniques [37]. Moreover, since our setting involves interactions with a rigid body, we import the mesh-based collider into AS-DiffMPM following the procedure described in Sec. 4.2. The *static* geometry of the continuum object is then reconstructed from the multi-view images at the initial timestep. With the scene setup complete, we initialize the physical parameters with an initial guess and start the optimization process.

Rendering Models. To the best of our knowledge, existing system identification methods [4, 7, 8, 9] do not account for scenarios involving continuum materials interacting with arbitrarily shaped rigid bodies. To address this limitation, we integrate AS-DiffMPM with several rendering models and establish a benchmark for this novel setting. Specifically, we combine AS-DiffMPM with the voxel-based NeRF from [5] (DVGO), the point-based 2D Gaussian Splatting method from [11] (2DGS), and the Motion-factorized Dynamic 3D Gaussian Splatting model (MDyn-3DGS) introduced in [8]. Unlike the first two methods, MDyn-3DGS performs dynamic scene reconstruction prior to optimization. This reconstructed information is then used during physical parameter optimization to constrain particles within the rendered dynamic scene.

Results. The results presented in Tab. 2 are summarized as follows:

- *Newtonian fluid:* MDyn-3DGS [8] generally provides more accurate estimates of physical parameters compared to 2DGS [11] and DVGO [5]. It also matches or exceeds the performance

Material	Collider	Parameters	DVGO [5]	2DGS [11]	MDyn-3DGS [8]
Newtonian	Box	$\log_{10}(\mu)$	6.71 ± 4.4	6.68 ± 4.55	5.58 ± 3.25
		$\log_{10}(\kappa)$	120.94 ± 73.37	112.13 ± 85.89	40.07 ± 39.47
	Bunny	$\log_{10}(\mu)$	3.55 ± 4.36	7.37 ± 5.91	3.29 ± 2.08
		$\log_{10}(\kappa)$	53.34 ± 64.94	99.16 ± 68.65	22.38 ± 17.32
	Armadillo	$\log_{10}(\mu)$	1.99 ± 1.59	5.28 ± 2.55	0.97 ± 0.86
		$\log_{10}(\kappa)$	32.94 ± 42.24	49.66 ± 35.79	14.99 ± 17.13
Non-Newtonian	Box	$\log_{10}(\mu)$	23.83 ± 14.77	28.88 ± 50.41	67.55 ± 128.49
		$\log_{10}(\kappa)$	52.76 ± 51.68	70.78 ± 88.13	32.10 ± 49.97
		$\log_{10}(\tau_Y)$	5.84 ± 4.34	12.77 ± 16.22	8.8 ± 2.86
		$\log_{10}(\eta)$	26.10 ± 19.21	58.02 ± 41.46	51.25 ± 28.81
	Bunny	$\log_{10}(\mu)$	55.38 ± 31.85	10.05 ± 7.95	15.91 ± 25.49
		$\log_{10}(\kappa)$	68.84 ± 38.11	44.44 ± 33.39	23.16 ± 18.29
		$\log_{10}(\tau_Y)$	11.43 ± 6.25	3.30 ± 2.15	4.97 ± 5.7
		$\log_{10}(\eta)$	62.06 ± 49.71	46.05 ± 27.02	43.37 ± 22.43
	Armadillo	$\log_{10}(\mu)$	79.23 ± 82.24	12.34 ± 13.32	12.62 ± 10.30
		$\log_{10}(\kappa)$	107.90 ± 95.32	33.30 ± 20.60	19.75 ± 13.67
		$\log_{10}(\tau_Y)$	63.01 ± 148.55	2.34 ± 2.41	4.32 ± 1.82
		$\log_{10}(\eta)$	83.60 ± 50.2	55.18 ± 37.20	39.74 ± 23.07
Granular	Box	θ_{fric}	4.16 ± 0.76	0.62 ± 0.75	3.16 ± 1.04
	Bunny	θ_{fric}	4.02 ± 0.68	0.50 ± 0.29	3.33 ± 0.62
	Armadillo	θ_{fric}	4.11 ± 1.05	0.36 ± 0.13	2.89 ± 0.78

Table 2: **System identification from visual observations.** We evaluate our AS-DiffMPM combined with three rendering methods on the task of physical parameter estimation from multi-view videos. Best (light red) and second best (very light red) results are highlighted.

obtained using particle trajectories as supervision (see Tab. 1), demonstrating that visual supervision provides meaningful additional information for system identification [4, 8].

- *Non-Newtonian fluid*: Consistent with prior work [4, 8], this material presents the greatest challenges, and no single rendering method consistently outperforms others across all colliders.
- *Granular*: lower absolute errors are generally observed for this material, indicating relatively simpler deformation behavior compared to fluids. Despite underperforming on fluid simulations, the explicit point-based representation of 2DGS [11] achieves superior results for granular media without relying on additional dynamic reconstruction supervision, as required by MDyn-3DGS [8].

Overall, MDyn-3DGS [8] typically outperforms 2DGS [11] and DVGO [5], highlighting the advantages of incorporating dynamic reconstruction into the rendering pipeline. However, the choice of method may ultimately depend on the complexity of the specific material-collider interactions being modeled. Finally, while prior work [4, 8] has shown promising results for system identification with planar surfaces, our findings indicate that handling more complex interactions is inherently more challenging—yet essential for advancing toward models that operate under fewer assumptions.

6 Real-World Application: Parameter Estimation of a Dough Sample

To further validate the proposed framework, we conduct a real-world experiment using a dough sample approximately the size of a football. The dough is modeled as a non-Newtonian material due to its viscoelastic behavior, which combines solid-like elasticity with fluid-like viscosity.

Experimental Setup. The dough is released in free fall and collides with the edge of a box, creating a non-planar contact scenario. Upon impact, part of the dough adheres to the horizontal surface, while the rest deforms and gradually flows along the vertical edge. The experiment is recorded using five cameras placed around the scene, and in each frame the dough is segmented using SAM2 [38].

Experiment Pipeline. The procedure consists of three stages:

1. **Static geometry reconstruction.** We reconstruct the scene without the dough in view using 2DGS [11] and import it into AS-DiffMPM as a collider. Then, we reconstruct the initial dough geometry using the multi-view images of the first timestep. We use isotropic Gaussians to ensure an even distribution along the surface. To avoid structural collapse upon collision during the simulation, we apply the internal infilling strategy from [1]. The resulting Gaussians are then fine-tuned, with densification and pruning disabled to preserve their distribution. These adjustments lead to more realistic MPM simulations.
2. **Initial motion estimation.** We estimate the initial velocity and gravity vectors from the free-fall phase and keep them fixed in subsequent stage.
3. **System identification.** With geometry and motion initialization in place, we optimize the dough’s physical parameters over the full sequence.

Results. The estimated physical parameters are: $\mu = 57,033.1$, $\kappa = 139,650.4$, $\tau_Y = 7,919.1$, $\eta = 25.1$. We further assess visual reconstruction quality over three seconds after collision. Tab. 3 shows the metrics averaged over 60 frames and five cameras. We observe a degradation in reconstruction quality in the later frames (see also Fig. 4). As the dough undergoes significant deformation, the Gaussians deviate from their original positions, introducing visual artifacts. A potential remedy is to incorporate *time-dependent Gaussian attributes*, which we leave for future work.

Time after collision	PSNR ($\uparrow$)	SSIM ($\uparrow$)	LPIPS ($\downarrow$)
1 s	33.2 ± 4.1	0.994 ± 0.004	0.019 ± 0.001
2 s	28.9 ± 6.6	0.988 ± 0.011	0.024 ± 0.011
3 s	25.3 ± 7.8	0.966 ± 0.013	0.035 ± 0.012

Table 3: Evaluation of image reconstruction quality for the dough over time.

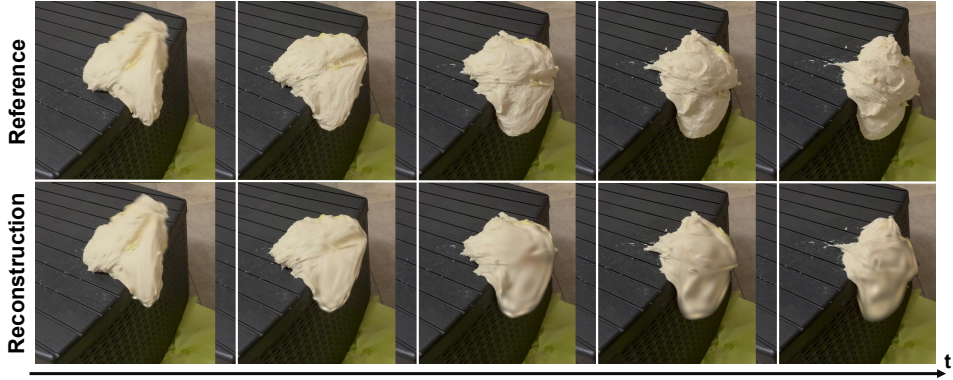


Figure 4: Example visualization of the dough experiment.

7 Conclusion

Limitation. One limitation of our work is the preliminary rigid body reconstruction-and-import into AS-DiffMPM, prior to start the system identification optimization. This restricts our approach to static colliders. Although the extension to moving rigid bodies is trivial in our framework, we leave the system identification analysis under such conditions as future work. Indeed, as significant deformation of the continuum material may occur, a carefully designed dynamic reconstruction method—similar to the one proposed in [8]—is required to mitigate visual artifacts.

In conclusion, we proposed AS-DiffMPM, a differentiable Material Point Method supporting arbitrarily shaped colliders. We validated our method quantitatively and combined it with various rendering models to perform system identification from visual observations, providing extensive results in a novel scenario where the continuum material undergoes deformations against complex colliders rather than planar surfaces.

References

- [1] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4389–4398, 2024.
- [2] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.
- [3] Deborah Sulsky, Shi-Jian Zhou, and Howard L. Schreyer. Application of a particle-in-cell method to solid mechanics. *Computer Physics Communications*, 87(1):236–252, 1995. Particle Simulation Methods.
- [4] Xuan Li, Yi-Ling Qiao, Peter Yichen Chen, Krishna Murthy Jatavallabhula, Ming Lin, Chenfanfu Jiang, and Chuang Gan. Pac-nerf: Physics augmented continuum neural radiance fields for geometry-agnostic system identification. In *International Conference on Learning Representations (ICLR)*, 2023.
- [5] Cheng Sun, Min Sun, and Hwann-Tzong Chen. 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, 2022.
- [6] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. DiffTaichi: Differentiable programming for physical simulation. In *International Conference on Learning Representations*, 2020.
- [7] Takuhiro Kaneko. Improving physics-augmented continuum neural radiance field-based geometry-agnostic system identification with lagrangian particle optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5480, 2024.
- [8] Junhao Cai, Yuji Yang, Weihao Yuan, HE Yisheng, Zilong Dong, Liefeng Bo, Hui Cheng, and Qifeng Chen. Gic: Gaussian-informed continuum for physical property identification and simulation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
- [9] Licheng Zhong, Hong-Xing Yu, Jiajun Wu, and Yunzhu Li. Reconstruction and simulation of elastic objects with spring-mass 3d gaussians. In *European Conference on Computer Vision*, pages 407–423. Springer, 2024.
- [10] Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.
- [11] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 conference papers*, pages 1–11, 2024.
- [12] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, 2011.
- [13] Derek Hoiem, Alexei A Efros, and Martial Hebert. Recovering surface layout from an image. *International Journal of Computer Vision*, 75:151–172, 2007.
- [14] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016.
- [15] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

- [16] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 2022.
- [17] Ayaan Haque, Matthew Tancik, Alexei A Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. In *ICCV*, 2023.
- [18] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew J Davison. In-place scene labelling and understanding with implicit scene representation. In *ICCV*, 2021.
- [19] Clément Jambon, Bernhard Kerbl, Georgios Kopanas, Stavros Diolatzis, George Drettakis, and Thomas Leimkühler. Nerfshop: Interactive editing of neural radiance fields. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 2023.
- [20] Ri-Zhao Qiu, Ge Yang, Weijia Zeng, and Xiaolong Wang. Language-driven physics-based scene synthesis and editing via feature splatting. In *European Conference on Computer Vision*, pages 368–383. Springer, 2024.
- [21] Yutao Feng, Xiang Feng, Yintong Shang, Ying Jiang, Chang Yu, Zeshun Zong, Tianjia Shao, Hongzhi Wu, Kun Zhou, Chenfanfu Jiang, et al. Gaussian splashing: Dynamic fluid synthesis with gaussian splatting. *arXiv e-prints*, pages arXiv–2401, 2024.
- [22] Ying Jiang, Chang Yu, Tianyi Xie, Xuan Li, Yutao Feng, Huamin Wang, Minchen Li, Henry Lau, Feng Gao, Yin Yang, et al. Vr-gs: A physical dynamics-aware interactive gaussian splatting system in virtual reality. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–1, 2024.
- [23] Yutao Feng, Yintong Shang, Xuan Li, Tianjia Shao, Chenfanfu Jiang, and Yin Yang. Pie-nerf: Physics-based interactive elastodynamics with nerf. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4450–4461, 2024.
- [24] Vismay Modi, Nicholas Sharp, Or Perel, Shinjiro Sueda, and David IW Levin. Simplicits: Mesh-free, geometry-agnostic elastic simulation. *ACM Transactions on Graphics (TOG)*, 43(4):1–11, 2024.
- [25] Piotr Borycki, Weronika Smolak, Joanna Waczyńska, Marcin Mazur, Sławomir Tadeja, and Przemysław Spurek. Gasp: Gaussian splatting for physic-based simulations. *arXiv preprint arXiv:2409.05819*, 2024.
- [26] Tianyuan Zhang, Hong-Xing Yu, Rundi Wu, Brandon Y Feng, Changxi Zheng, Noah Snively, Jiajun Wu, and William T Freeman. Physdreamer: Physics-based interaction with 3d objects via video generation. In *European Conference on Computer Vision*, pages 388–406. Springer, 2024.
- [27] Yuchen Lin, Chenguo Lin, Jianjin Xu, and Yadong MU. OmniphysGS: 3d constitutive gaussians for general physics-based dynamics generation. In *The Thirteenth International Conference on Learning Representations*, 2025.
- [28] Tianyu Huang, Haoze Zhang, Yihan Zeng, Zhilu Zhang, Hui Li, Wangmeng Zuo, and Rynson WH Lau. Dreamphysics: Learning physical properties of dynamic 3d gaussians with video diffusion priors. *arXiv preprint arXiv:2406.01476*, 2024.
- [29] Fangfu Liu, Hanyang Wang, Shunyu Yao, Shengjun Zhang, Jie Zhou, and Yueqi Duan. Physics3d: Learning physical properties of 3d gaussians via video diffusion. *arXiv preprint arXiv:2406.04338*, 2024.
- [30] Dimitris Metaxas and Demetri Terzopoulos. Shape and nonrigid motion estimation through physics-based synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(6):580–591, 1993.
- [31] Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. A material point method for snow simulation. *ACM Trans. Graph.*, 32(4), July 2013.
- [32] Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. The material point method for simulating continuum materials. In *ACM SIGGRAPH 2016 Courses*, SIGGRAPH ’16, New York, NY, USA, 2016. Association for Computing Machinery.

- [33] Eduardo WV Chaves. *Notes on continuum mechanics*. Springer Science & Business Media, 2013.
- [34] Michael Steffen, Robert M Kirby, and Martin Berzins. Analysis and reduction of quadrature errors in the material point method (mpm). *International journal for numerical methods in engineering*, 76(6):922–948, 2008.
- [35] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.
- [36] Pingchuan Ma, Peter Yichen Chen, Bolei Deng, Joshua B Tenenbaum, Tao Du, Chuang Gan, and Wojciech Matusik. Learning neural constitutive laws from motion observations for generalizable pde dynamics. In *International Conference on Machine Learning*, pages 23279–23300. PMLR, 2023.
- [37] Shanchuan Lin, Andrey Ryabtsev, Soumyadip Sengupta, Brian L Curless, Steven M Seitz, and Ira Kemelmacher-Shlizerman. Real-time high-resolution background matting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8762–8771, 2021.
- [38] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. In *International Conference on Learning Representations (ICLR)*, 2025.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: The main claims made in the abstract and introduction reflect the paper's contributions and objectives.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: The paper presents a Conclusion section where limitations are discussed.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[NA\]](#)

Justification: The paper does not contain theoretical results

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: We did our best to include all the details for reproducibility. Moreover, we will release the code of our framework.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: Instructions on how to download the data and run scripts are not included in the paper, however, we will rely on a project page (linked to our paper) to provide instructions for reproducibility. We will release ready-to-use data and scripts (e.g., no data pre-processing required).

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We both described our training/test details (in the Experiments section) and referred to prior works [4] when the settings are unchanged.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We reported tables for quantitative analysis in the Experiments section

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We described this in the supplementary material.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: This work conform with the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: There is no societal impact of the work performed.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: This paper poses no such risks

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The used assets are properly cited throughout the paper and more specifically in the Experiments section.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset’s creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: This paper does not release new assets

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core methodology and contributions of this paper do not involve LLM

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.

Appendix

This supplementary material provides additional results that both complement the main paper and preliminarily explore more scenarios. We invite the reader to also refer to the supplementary media attached to this paper and the project page for additional visualizations.

A Additional Results on System Identification

This section complements the results in Tab. 1 and Tab. 2 with additional quantitative metrics and qualitative visualizations.

A.1 System Identification from Particle Trajectories

A.1.1 Quantitative Results on Geometric Error

Beyond the results reported in Tab. 1, we quantify the geometric accuracy of the simulated particle trajectories (after convergence of the physical parameter optimization) by comparing them against the reference trajectory used for supervision. Following [8, 9], we report the Chamfer Distance (CD) and Earth Mover’s Distance (EMD). The Chamfer Distance is computed using the squared distance and is expressed in units of 10^3 mm^2 . Each metric is averaged over all timesteps across the dataset. Results are presented in Tab. 4

Material	Collider	Metrics	RP-DiffMPM	GOP-DiffMPM	AS-DiffMPM (Ours)
Newtonian	Box	CD ↓	3.903 ± 2.603	3.879 ± 2.625	3.925 ± 2.590
		EMD ↓	0.066 ± 0.018	0.066 ± 0.019	0.066 ± 0.019
	Bunny	CD ↓	2.948 ± 2.060	2.946 ± 2.059	2.955 ± 2.059
		EMD ↓	0.059 ± 0.017	0.059 ± 0.017	0.059 ± 0.017
	Armadillo	CD ↓	2.977 ± 2.515	2.964 ± 2.512	2.960 ± 2.521
		EMD ↓	0.064 ± 0.025	0.064 ± 0.025	0.064 ± 0.024
Non-Newtonian	Box	CD ↓	16.566 ± 6.847	16.492 ± 6.854	16.558 ± 6.868
		EMD ↓	0.108 ± 0.023	0.107 ± 0.023	0.107 ± 0.022
	Bunny	CD ↓	12.858 ± 5.323	12.822 ± 5.308	12.826 ± 5.353
		EMD ↓	0.098 ± 0.023	0.098 ± 0.023	0.098 ± 0.023
	Armadillo	CD ↓	15.576 ± 10.921	15.571 ± 10.909	15.519 ± 10.936
		EMD ↓	0.121 ± 0.045	0.121 ± 0.045	0.121 ± 0.045
Granular	Box	CD ↓	3.478 ± 1.054	3.485 ± 1.090	3.448 ± 1.043
		EMD ↓	0.064 ± 0.005	0.064 ± 0.006	0.063 ± 0.005
	Bunny	CD ↓	3.084 ± 0.820	3.067 ± 0.847	3.055 ± 0.851
		EMD ↓	0.058 ± 0.006	0.058 ± 0.005	0.058 ± 0.005
	Armadillo	CD ↓	2.896 ± 0.857	2.860 ± 0.845	2.858 ± 0.902
		EMD ↓	0.056 ± 0.006	0.056 ± 0.006	0.056 ± 0.006

Table 4: Evaluation of geometric error using CD and EMD for different materials and colliders.

A.1.2 Qualitative Visualizations

We provide example visualizations of the 3D particle trajectories used in our experiments (see Fig. 5). Please refer to the supplementary material attached to this paper for more videos.

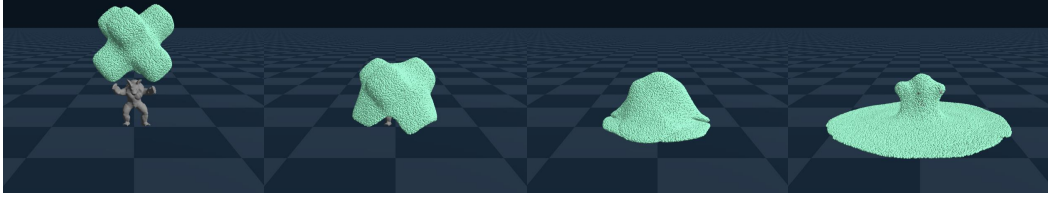


Figure 5: Particle trajectories of a Newtonian material colliding with the Armadillo rigid body.

A.2 System Identification from Visual Observations

A.2.1 Quantitative Results on Image Reconstruction

In addition to the results shown in Tab. 2, we assess the image reconstruction quality of all models after convergence of the physical parameter optimization. We report in Tab. 5 the standard metrics used in radiance field evaluation: Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM), and Learned Perceptual Image Patch Similarity (LPIPS). Each metric is averaged over all views and timesteps across the dataset.

Material	Collider	Metrics	DVGO [5]	2DGS [11]	MDyn-3DGS [8]
Newtonian	Box	PSNR $\uparrow$	35.318 ± 4.901	35.669 ± 6.779	41.300 ± 2.130
		SSIM $\uparrow$	0.985 ± 0.010	0.978 ± 0.022	0.997 ± 0.002
		LPIPS $\downarrow$	0.023 ± 0.016	0.022 ± 0.019	0.017 ± 0.007
	Bunny	PSNR $\uparrow$	36.491 ± 4.217	36.949 ± 6.083	41.537 ± 1.856
		SSIM $\uparrow$	0.987 ± 0.009	0.981 ± 0.019	0.998 ± 0.001
		LPIPS $\downarrow$	0.021 ± 0.014	0.019 ± 0.017	0.015 ± 0.006
	Armadillo	PSNR $\uparrow$	36.905 ± 3.409	37.484 ± 5.340	41.888 ± 1.395
		SSIM $\uparrow$	0.987 ± 0.008	0.982 ± 0.017	0.998 ± 0.001
		LPIPS $\downarrow$	0.021 ± 0.012	0.019 ± 0.015	0.015 ± 0.005
Non-Newtonian	Box	PSNR $\uparrow$	28.331 ± 5.462	29.468 ± 6.658	48.521 ± 2.291
		SSIM $\uparrow$	0.973 ± 0.014	0.980 ± 0.015	0.998 ± 0.001
		LPIPS $\downarrow$	0.037 ± 0.014	0.023 ± 0.015	0.011 ± 0.003
	Bunny	PSNR $\uparrow$	28.871 ± 5.077	31.718 ± 5.184	48.659 ± 2.278
		SSIM $\uparrow$	0.975 ± 0.012	0.986 ± 0.011	0.998 ± 0.001
		LPIPS $\downarrow$	0.035 ± 0.012	0.018 ± 0.011	0.010 ± 0.003
	Armadillo	PSNR $\uparrow$	27.196 ± 5.573	32.176 ± 4.459	48.762 ± 1.814
		SSIM $\uparrow$	0.971 ± 0.013	0.986 ± 0.009	0.998 ± 0.001
		LPIPS $\downarrow$	0.038 ± 0.014	0.018 ± 0.010	0.010 ± 0.002
Granular	Box	PSNR $\uparrow$	31.338 ± 5.558	31.055 ± 7.419	40.527 ± 3.951
		SSIM $\uparrow$	0.967 ± 0.023	0.957 ± 0.040	0.992 ± 0.006
		LPIPS $\downarrow$	0.046 ± 0.028	0.035 ± 0.029	0.022 ± 0.009
	Bunny	PSNR $\uparrow$	32.487 ± 4.841	32.063 ± 6.870	42.522 ± 2.127
		SSIM $\uparrow$	0.972 ± 0.018	0.963 ± 0.034	0.994 ± 0.003
		LPIPS $\downarrow$	0.042 ± 0.024	0.031 ± 0.026	0.018 ± 0.007
	Armadillo	PSNR $\uparrow$	32.204 ± 4.577	31.361 ± 6.936	41.900 ± 2.122
		SSIM $\uparrow$	0.970 ± 0.018	0.959 ± 0.035	0.993 ± 0.003
		LPIPS $\downarrow$	0.044 ± 0.024	0.034 ± 0.027	0.021 ± 0.008

Table 5: Evaluation of image reconstruction using PSNR, SSIM, and LPIPS for different materials and colliders.

A.2.2 Qualitative Visualizations

We provide example visualizations of the models at convergence (see Fig. 6 and Fig. 7). We remind that, prior to the physical parameters optimization, the rigid collider is imported into AS-DiffMPM for collision handling and the continuum material is segmented out from the ground truth views. During physical parameters optimization, the visual observations containing the segmented-out continuum object are used for supervision. Please refer to the supplementary material attached to this paper for videos, along with the reconstruction metrics and estimated physical parameters.

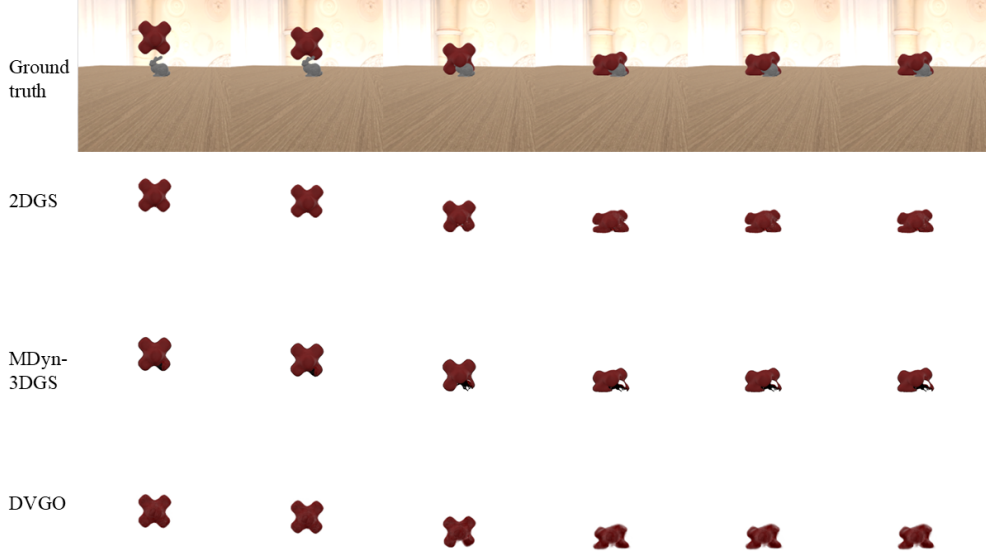


Figure 6: Non-Newtonian material colliding with the Bunny rigid body.

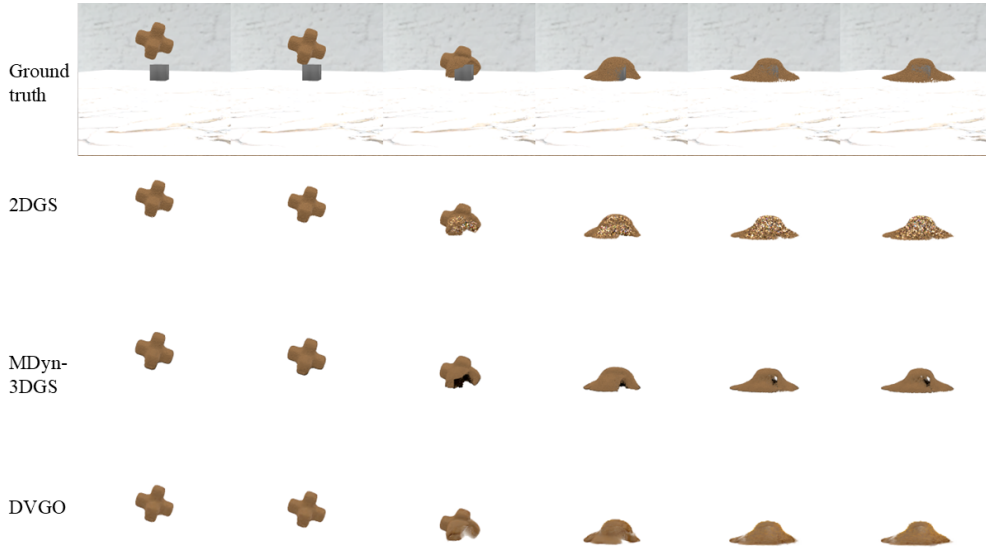


Figure 7: Sand material colliding with the Box rigid body.

B Mesh vs. 2DGS as Rigid Body Collider

In contrast to the experiments presented in the main paper (Sec.5), where the collider imported into AS-DiffMPM is represented as a mesh, in this analysis we segment and reconstruct the collider using 2DGS[11], and import it into AS-DiffMPM as described in Sec. 4.2. We then perform the same system identification experiment based on particle trajectories to evaluate the impact of using a mesh compared to 2DGS as rigid body collider representation. Results are presented in Tab. 6 and Tab. 7. As expected, 2DGS-based colliders obtains lower accuracy. This is due to the reconstructed Gaussians not being as precise as a mesh, which introduce noise into the Collision grid (Sec. 4.1.1) and result in less accurate collision handling. However, we remark that our AS-DiffMPM is versatile enough to support both mesh and 2DGS-based colliders. This flexibility is particularly valuable for real-world scenarios, where colliders may need to be reconstructed on-the-fly using 2DGS to enable simulation.

Material	Collider	Parameters	AS-DiffMPM w/ 2DGS	AS-DiffMPM w/ Mesh
Newtonian	Box	$\log_{10}(\mu)$	4.36 ± 2.66	5.08 ± 2.73
		$\log_{10}(\kappa)$	36.26 ± 24.63	27.81 ± 33.49
	Bunny	$\log_{10}(\mu)$	7.03 ± 4.09	4.35 ± 5.24
		$\log_{10}(\kappa)$	81.30 ± 31.53	31.37 ± 50.02
	Armadillo	$\log_{10}(\mu)$	8.75 ± 6.17	0.4 ± 0.76
		$\log_{10}(\kappa)$	86.74 ± 47.68	0.22 ± 0.24
Non-Newtonian	Box	$\log_{10}(\mu)$	55.51 ± 42.28	23.51 ± 30.07
		$\log_{10}(\kappa)$	126.10 ± 99.60	27.06 ± 19.12
		$\log_{10}(\tau_Y)$	11.18 ± 8.63	7.28 ± 3.75
		$\log_{10}(\eta)$	56.71 ± 35.83	17.7 ± 15.17
	Bunny	$\log_{10}(\mu)$	53.12 ± 44.90	21.92 ± 36.89
		$\log_{10}(\kappa)$	110.70 ± 73.58	26.77 ± 25.09
		$\log_{10}(\tau_Y)$	16.40 ± 5.74	2.48 ± 1.87
		$\log_{10}(\eta)$	56.57 ± 35.17	39.1 ± 36.2
	Armadillo	$\log_{10}(\mu)$	25.22 ± 24.02	7.14 ± 6.64
		$\log_{10}(\kappa)$	124.88 ± 75.89	56.26 ± 76.97
		$\log_{10}(\tau_Y)$	28.43 ± 19.45	5.07 ± 5.79
		$\log_{10}(\eta)$	51.76 ± 30.75	44.98 ± 41.34
Granular	Box	θ_{fric}	1.86 ± 1.14	1.22 ± 0.71
	Bunny	θ_{fric}	0.91 ± 1.15	0.06 ± 0.07
	Armadillo	θ_{fric}	1.38 ± 1.08	0.51 ± 0.96

Table 6: **System identification from particle trajectories.** Evaluation (reported as absolute error) of physical parameter estimation when using mesh and 2DGS [11] as collider.

Material	Collider	Metrics	AS-DiffMPM w/ 2DGS	AS-DiffMPM w/ Mesh
Newtonian	Box	CD ↓	3.893 ± 2.586	3.925 ± 2.590
		EMD ↓	0.066 ± 0.018	0.066 ± 0.019
	Bunny	CD ↓	2.955 ± 2.049	2.937 ± 2.059
		EMD ↓	0.059 ± 0.017	0.059 ± 0.017
	Armadillo	CD ↓	2.962 ± 2.515	2.960 ± 2.521
		EMD ↓	0.064 ± 0.024	0.064 ± 0.024
Non-Newtonian	Box	CD ↓	16.629 ± 6.882	16.558 ± 6.868
		EMD ↓	0.108 ± 0.023	0.107 ± 0.022
	Bunny	CD ↓	12.867 ± 5.355	12.826 ± 5.353
		EMD ↓	0.098 ± 0.023	0.098 ± 0.023
	Armadillo	CD ↓	15.538 ± 10.854	15.519 ± 10.936
		EMD ↓	0.121 ± 0.045	0.121 ± 0.045
Granular	Box	CD ↓	3.468 ± 1.042	3.448 ± 1.043
		EMD ↓	0.064 ± 0.005	0.063 ± 0.005
	Bunny	CD ↓	3.094 ± 0.867	3.055 ± 0.851
		EMD ↓	0.057 ± 0.005	0.058 ± 0.005
	Armadillo	CD ↓	2.872 ± 0.886	2.858 ± 0.902
		EMD ↓	0.056 ± 0.006	0.056 ± 0.006

Table 7: **System identification from particle trajectories.** Evaluation of geometric error when using mesh or 2DGS as collider.

C Exploring More Conditions for System Identification

We further benchmark our AS-DiffMPM framework under more conditions, both in terms of colliders and materials. This section is not intended to provide an in-depth analysis for each condition, but rather to explore a wide range of possibilities, aiming to possibly identify which ones deserve more thorough analysis in future work. All experiments are conducted using the pipeline described in Sec. 5.1 for system identification from particle trajectories.

C.1 Same Colliders in Different Conditions

We use the three colliders (Box, Bunny and Armadillo) of our analysis in the main paper but changing characteristics. For each case, results are reported per material type, averaging over the colliders (see Tab. 8)

- **Condition 1.** The collider is moved to the side (instead of being underneath the object) such that the continuum material first impacts the ground and then interacts with the collider as it spreads. We perform nine experiments—comprising three material types and three colliders.
- **Condition 2.** Each collider is tested at two scales (0.5 and 1.5), for all material types and colliders, resulting in 18 experiments.
- **Condition 3.** We combine all the colliders at once, placing them side by side, so that the material interacts with all of them upon collision. This results in three experiments, one per material type

Overall, Condition 1 appears to be the one with more significant challenges in physical parameter estimation, likely due to this ground-first-collider-next complex interactions between the material and the collider.

Material	Parameter	Cond. 1	Cond. 2	Cond. 3
Newtonian	$\log_{10}(\mu)$	10.51 ± 12.09	5.21 ± 7.02	6.85
	$\log_{10}(\kappa)$	39.82 ± 9.65	26.88 ± 25.91	47.34
Non-Newtonian	$\log_{10}(\mu)$	23.02 ± 10.31	22.17 ± 12.81	3.41
	$\log_{10}(\kappa)$	50.34 ± 7.78	23.15 ± 15.64	41.98
	$\log_{10}(\tau_Y)$	9.56 ± 5.90	5.34 ± 4.26	7.59
	$\log_{10}(\eta)$	34.41 ± 12.74	21.03 ± 7.98	32.02
Granular	θ_{fric}	4.02 ± 1.70	1.09 ± 0.93	3.04

Table 8: System identification errors across three conditions.

Material	Parameter	S-MPM	RP-DiffMPM	GOP-DiffMPM	AS-DiffMPM (Ours)
Newtonian	$\log_{10}(\mu)$	7.51	8.03	7.63	7.98
	$\log_{10}(\kappa)$	31.98	35.21	32.45	33.21
Non-Newtonian	$\log_{10}(\mu)$	24.09	24.57	24.91	24.03
	$\log_{10}(\kappa)$	19.01	21.12	20.19	19.03
	$\log_{10}(\tau_Y)$	4.91	5.12	4.78	4.54
	$\log_{10}(\eta)$	30.53	32.10	31.05	31.21
Granular	θ_{fric}	0.15	0.56	0.22	0.12

Table 9: System identification errors for planar collider.

C.2 More Colliders and Material Types

In this section, we focus both on different colliders and material types, comparing our AS-DiffMPM framework with baselines. For the former, we first explore a comparison using the condition adopted in previous works [4, 7, 8, 9], i.e., a planar collider, and then with a bowl-shaped collider. Instead, for the latter, we explore both further material types and shapes.

Planar Collider. We conduct experiments comparing our AS-DiffMPM with both the baselines used in our main paper (GOP-DiffMPM and RP-DiffMPM) and the Standard MPM (S-MPM) collision strategy adopted by previous works [4, 7, 8, 9]. In S-MPM, collisions are resolved at the Grid Operations stage (Sec. 3.1): for a planar surface defined by point $\mathbf{x}_b$ and normal $\mathbf{n}_b$, the nodal velocity $\mathbf{v}_g$ is set to zero for all grid nodes $\mathbf{x}_g$ such that $(\mathbf{x}_g - \mathbf{x}_b) \cdot \mathbf{n}_b \leq 0$. In contrast, AS-DiffMPM represents the same planar collider as a mesh surface positioned at $\mathbf{x}_b$, and applies particle-wise collision handling during both P2G and G2P steps. We performed three experiments, one per material type (see Tab. 9). The S-MPM achieves slightly lower errors for some parameters, likely due to its use of analytically defined boundary conditions, which may offer improved numerical stability in simple geometries.

Bowl Collider. We import a bowl-shaped collider into our framework and place it such that the falling material impacts its open surface—causing some particles to enter the bowl while others fall outside. We conducted three experiments, one per material type (see Tab. 10). The general trend observed in the main paper appears to be consistent here, with our method outperforming the baselines.

More material types. While the main focus of our work is on Newtonian, non-Newtonian and granular materials, we explore here also with elastic and plasticine materials. For elastic material, we consider Young’s modulus (E) and Poisson’s ration (ν), while for plasticine material, we consider also yield stress (τ_Y). We refer the reader to [4] for the constitutive models. Each method was evaluated on six experiments: two material types and with three colliders. The results are presented in Tab. 11, averaged over the colliders. Interestingly, all methods generally perform well on the elastic material. This may be due to the simpler nature of the interaction between the elastic material and the collider—since the material tends to bounce away upon contact, the particle-wise collision handling mechanism of AS-DiffMPM has less impact. Indeed, in these scenarios, no fine-grained interaction (e.g., material flowing around sharp corners) occurs.

Material	Parameter	RP-DiffMPM	GOP-DiffMPM	AS-DiffMPM (Ours)
Newtonian	$\log_{10}(\mu)$	7.84	7.29	6.01
	$\log_{10}(\kappa)$	30.12	28.45	25.67
Non-Newtonian	$\log_{10}(\mu)$	19.31	20.05	15.03
	$\log_{10}(\kappa)$	38.22	35.79	29.84
	$\log_{10}(\tau_Y)$	7.31	6.98	5.15
	$\log_{10}(\eta)$	26.78	28.45	20.94
Granular	θ_{fric}	3.01	1.17	1.41

Table 10: System identification errors for bowl collider.

Material	Parameter	RP-DiffMPM	GOP-DiffMPM	AS-DiffMPM (Ours)
Elastic	$\log_{10}(E)$	2.01 ± 0.45	2.87 ± 1.18	1.91 ± 0.96
	ν	0.36 ± 0.43	1.23 ± 0.32	0.49 ± 0.12
Plasticine	$\log_{10}(E)$	28.45 ± 3.12	16.88 ± 1.84	12.07 ± 2.21
	ν	9.29 ± 5.04	10.24 ± 3.03	8.21 ± 1.02
	$\log_{10}(\tau_Y)$	9.44 ± 2.05	8.76 ± 1.88	6.23 ± 1.47

Table 11: System identification errors for elastic and plasticine material types.

Shape	Material	Parameter	RP-DiffMPM	GOP-DiffMPM	AS-DiffMPM (Ours)
Droplet	Newtonian	$\log_{10}(\mu)$	7.81 ± 3.52	6.94 ± 4.41	5.87 ± 3.37
		$\log_{10}(\kappa)$	32.33 ± 3.25	21.91 ± 8.13	24.50 ± 3.98
Letter	Newtonian	$\log_{10}(\mu)$	8.12 ± 3.49	7.22 ± 2.38	6.05 ± 1.35
		$\log_{10}(\kappa)$	31.01 ± 5.40	23.12 ± 2.17	22.36 ± 6.04
Cream	Non-Newtonian	$\log_{10}(\mu)$	19.42 ± 7.61	18.67 ± 2.53	14.73 ± 5.12
		$\log_{10}(\kappa)$	46.50 ± 12.09	44.12 ± 3.88	38.08 ± 4.65
		$\log_{10}(\tau_Y)$	6.92 ± 0.58	6.58 ± 0.47	4.97 ± 0.34
		$\log_{10}(\eta)$	54.88 ± 12.72	46.41 ± 5.85	40.01 ± 7.43
Toothpaste	Non-Newtonian	$\log_{10}(\mu)$	20.05 ± 5.45	19.18 ± 1.29	15.30 ± 1.06
		$\log_{10}(\kappa)$	44.44 ± 4.33	44.99 ± 3.04	39.72 ± 3.89
		$\log_{10}(\tau_Y)$	7.35 ± 1.61	6.97 ± 0.52	5.12 ± 0.38
		$\log_{10}(\eta)$	62.19 ± 19.06	47.55 ± 21.95	41.17 ± 9.01
Torus	Elastic	$\log_{10}(E)$	1.85 ± 0.83	2.87 ± 0.75	2.69 ± 0.62
		ν	0.22 ± 0.02	0.25 ± 0.02	0.19 ± 0.01
Bird	Elastic	$\log_{10}(E)$	2.31 ± 0.90	2.94 ± 0.81	2.51 ± 0.69
		ν	0.22 ± 0.02	0.26 ± 0.02	0.21 ± 0.01
Playdoh	Plasticine	$\log_{10}(E)$	17.38 ± 1.14	15.89 ± 1.03	11.62 ± 0.89
		ν	8.31 ± 3.03	9.27 ± 2.02	6.22 ± 2.01
		$\log_{10}(\tau_Y)$	9.91 ± 1.46	8.70 ± 0.62	6.74 ± 0.51
Cat	Plasticine	$\log_{10}(E)$	18.11 ± 1.20	16.32 ± 1.05	12.01 ± 0.95
		ν	8.30 ± 2.71	8.28 ± 1.02	6.23 ± 1.98
		$\log_{10}(\tau_Y)$	10.02 ± 0.84	9.14 ± 0.71	7.20 ± 0.57
Trophy	Granular	θ_{fric}	2.19 ± 0.15	1.95 ± 0.12	1.24 ± 0.09

Table 12: System identification errors for different material shapes.

More material shapes. We use the 9 shapes and their associated material types introduced in [4]. Each method is tested on all 9 shapes, with each shape colliding against the three colliders, resulting in 27 experiments per method. The results are presented in Tab. 12, averaged over the colliders.

D Extensions of AS-DiffMPM

Due to our collision handling mechanism integrated into the *P2G* and *G2P* stages, the AS-DiffMPM framework can accommodate additional physical constraints. In the following, we present two illustrative examples.

D.1 Material Cutting

We consider a thin surface intersecting with a continuum material. We implement a *separating surface* behavior where we adjust the velocity of the particle $\mathbf{v}_p$ based on its relative position to the surface. Concretely, during the *G2P* stage, for each incompatible particle (Sec. 4.1.2), we compute the *signed* particle normal $\bar{\mathbf{n}}_p = \mathbf{n}_p T_p$, and update the velocity according to the particle’s motion relative to the surface. If the particle is moving away from the boundary (i.e., $\mathbf{v}_p \cdot \bar{\mathbf{n}}_p > 0$), its velocity remains unchanged. Otherwise, we project the velocity along the surface: $\mathbf{v}_p^{proj} = \mathbf{v}_p - (\mathbf{v}_p \cdot \bar{\mathbf{n}}_p) \bar{\mathbf{n}}_p$.

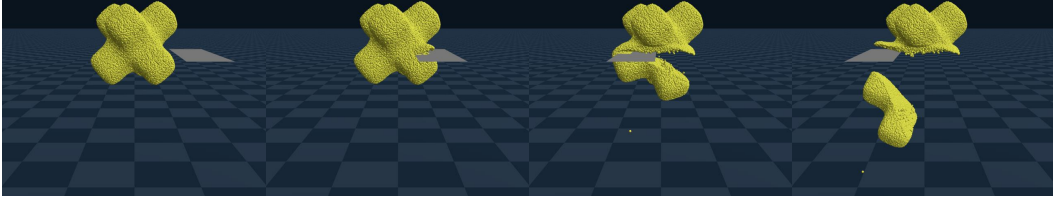


Figure 8: A thin surface cutting an elastic object.

D.2 Collider-to-Continuum Coupling

We present a scenario where a moving collider can interact with the continuum material. During the *G2P* stage, for each incompatible particle, we correct the velocity as $\bar{\mathbf{v}}_p = \mathbf{v}_p + \mathbf{v}_r$, where $\mathbf{v}_r$ is the rigid body velocity at position $\mathbf{x}_p$.

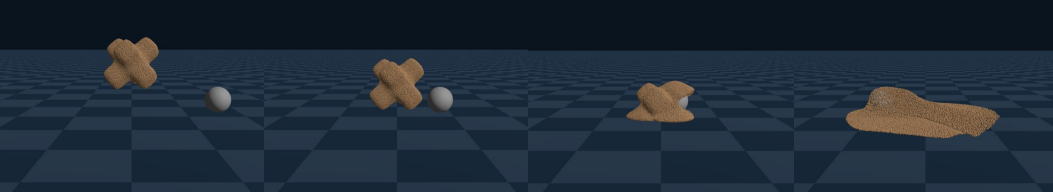


Figure 9: A ball colliding with a sand object.

E Values of Physical Parameters for Ground-truth Rollouts

For a fair comparison, we generate ground-truth simulation rollouts using the same physical parameter values as in [4], listed in Tab. 13 for convenience. Each ground-truth rollout is executed three times, one for each collider, totaling 75 rollouts. We remark that for each rollout a separate training run of system identification is performed, where an initial guess of the physical parameters is optimized by comparing the current rollout against the reference (either in the form of particle trajectories or visual observations). For each material type, we use the same initial guess across all training runs.

Newtonian (Initial Guess: $\mu = 10, \kappa = 10^4$)										
Parameter	1	2	3	4	5	6	7	8	9	10
μ	19.46	436.62	155.83	121.76	49.09	38.44	64.16	228.71	552.98	106.93
κ	56075.55	152696.25	193525.59	257356.05	518012.47	13772.52	358237.13	11041.06	16789.77	112569.73
Non-Newtonian (Initial Guess: $\mu = 100, \kappa = 10^5, \tau_Y = 10, \eta = 1$)										
Parameter	1	2	3	4	5	6	7	8	9	10
μ	13209.25	65351.08	43757.04	36027.61	19593.71	20522.72	51549.45	121865.90	241579.97	33764.59
κ	201566.59	171054.03	249639.94	134751.55	121836.33	14494.30	370317.66	32859.59	30324.98	122896.10
τ_Y	1151.42	7491.70	3964.94	5061.12	1462.78	4153.38	3203.67	1192.76	1251.29	4689.16
η	6.68	26.69	23.27	22.31	38.83	27.24	20.43	10.27	10.62	22.89
Granular (Initial Guess: $\theta_{fric} = 10$)										
Parameter	1	2	3	4	5					
θ_{fric}	30.6577	32.3751	26.8816	29.3458	42.2861					

Table 13: Values of physical parameters for ground-truth simulation rollouts.

F Implementation Details

We implemented our framework using Python and Taichi for differentiable programming and parallel computation. AS-DiffMPM is built upon the open source DiffMPM implementation in [4] and subsequent works [7, 8]. We run the experiments on NVIDIA RTX 3080 and 4090 graphics cards.