PhysGaussian: Physics-Integrated 3D Gaussians for Generative Dynamics

Tianyi Xie[1*] Zeshun Zong[1*] Yuxing Qiu[1*] Xuan Li[1*]
Yutao Feng[2,3] Yin Yang[3] Chenfanfu Jiang[1]
[1] UCLA, [2] Zhejiang University, [3] University of Utah

Figure 1. **PhysGaussian** is a unified simulation-rendering pipeline based on 3D Gaussians and continuum mechanics.

Abstract

We introduce PhysGaussian, a new method that seamlessly integrates physically grounded Newtonian dynamics within 3D Gaussians to achieve high-quality novel motion synthesis. Employing a custom Material Point Method (MPM), our approach enriches 3D Gaussian kernels with physically meaningful kinematic deformation and mechanical stress attributes, all evolved in line with continuum mechanics principles. A defining characteristic of our method is the seamless integration between physical simulation and visual rendering: both components utilize the same 3D Gaussian kernels as their discrete representations. This negates the necessity for triangle/tetrahedron meshing, marching cubes, "cage meshes," or any other geometry embedding, highlighting the principle of "what you see is what you simulate (WS^2)." Our method demonstrates exceptional versatility across a wide variety of materials–including elastic entities, plastic metals, non-Newtonian fluids, and granular materials–showcasing its strong capabilities in creating diverse visual content with novel viewpoints and movements. Our project page is at: https://xpandora.github.io/PhysGaussian/.

1. Introduction

Recent strides in Neural Radiance Fields (NeRFs) have showcased significant advancements in 3D graphics and vision [24]. Such gains have been further augmented by the cutting-edge 3D Gaussian Splatting (GS) framework [16]. 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 [12, 28, 47, 51].

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, employed for both simulation and rendering. In essence, our approach champions the principle of *"what you see is what you simulate" (WS^2)* [25], 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.

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 [24]. 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 [1, 7, 20, 26, 40, 46]. 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 [16]. 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. [30] and Park et al. [27] 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 [5, 19, 21, 28, 31, 32, 51]. Additionally, Liu et al. [21], Qiao et al. [31], Yuan et al. [51] 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 [16, 46]. In particular, Chen et al. [6], Luiten et al. [22], Wu et al. [45], Yang et al. [48, 49] directly manipulate NeRF fields. Li et al. [18] 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 [10]. 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, sand, and snow [13, 17, 39]. MPM can also be expanded to simulate objects that possess codimensional characteristics [15]. In addition, the efficacy of utilizing GPU(s)

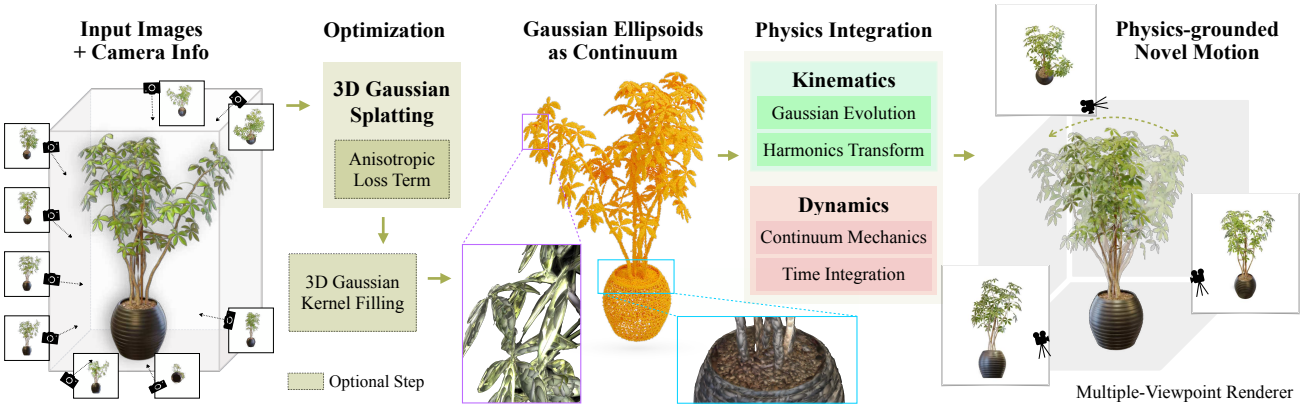


Figure 2. **Method Overview.** PhysGaussian is a unified simulation-rendering pipeline that incorporates 3D Gaussian splatting representation and continuum mechanics to generate physics-based dynamics and photo-realistic renderings simultaneously and seamlessly.

to accelerate MPM implementations has also been demonstrated in [8, 11, 33, 44]. 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.

3. Method Overview

We propose PhysGaussian (Fig. 2), a unified simulation-rendering framework for generative dynamics based on continuum mechanics and 3D GS. Adopted from Kerbl et al. [16], 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.

3.1. 3D Gaussian Splatting

3D Gaussian Splatting method [16] reparameterizes NeRF [24] 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

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

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 [45] and minimizing rendering losses over videos. In Sec. 3.1, we show that this time-dependent evolution can be given by the continuum deformation map.

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 [2]. 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:

$$\int_{B_\epsilon^t} \rho(\boldsymbol{x}, t) \equiv \int_{B_\epsilon^0} \rho(\boldsymbol{\phi}^{-1}(\boldsymbol{x}, t), 0), \tag{2}$$

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

$$\rho(\boldsymbol{x}, t)\dot{\boldsymbol{v}}(\boldsymbol{x}, t) = \nabla \cdot \boldsymbol{\sigma}(\boldsymbol{x}, t) + \boldsymbol{f}^{\text{ext}}, \tag{3}$$

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 [2, 14]. 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 [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 [14, 39]. 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. [39] 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

$$\frac{m_i}{\Delta t}(\boldsymbol{v}_i^{n+1} - \boldsymbol{v}_i^n) = -\sum_p V_p^0 \frac{\partial \Psi}{\partial \boldsymbol{F}}(\boldsymbol{F}_p^{E,n})\boldsymbol{F}_p^{E,n\,T}\nabla w_{ip}^n + \boldsymbol{f}_i^{ext}. \quad (4)$$

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 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 $\boldsymbol{F}^P$ [37], 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.

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 $\boldsymbol{\phi}(\boldsymbol{X}, t)$,

$$G_p(\boldsymbol{x}, t) = e^{-\frac{1}{2}(\boldsymbol{\phi}^{-1}(\boldsymbol{x},t)-\boldsymbol{X}_p)^T \boldsymbol{A}_p^{-1}(\boldsymbol{\phi}^{-1}(\boldsymbol{x},t)-\boldsymbol{X}_p)} \quad (5)$$

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

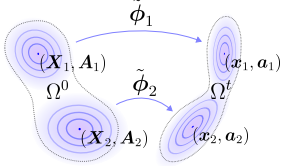
$$\tilde{\phi}_p(\boldsymbol{X}, t) = \boldsymbol{x}_p + \boldsymbol{F}_p(\boldsymbol{X} - \boldsymbol{X}_p), \quad (6)$$

the deformed kernel becomes Gaussian as desired:

$$G_p(\boldsymbol{x}, t) = e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}_p)^T (\boldsymbol{F}_p \boldsymbol{A}_p \boldsymbol{F}_p^T)^{-1}(\boldsymbol{x}-\boldsymbol{x}_p)}. \quad (7)$$

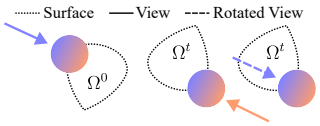
This transformation naturally provides a time-dependent version of $\boldsymbol{x}_p$ and $\boldsymbol{A}_p$ for the 3D GS framework:

$$\begin{aligned} \boldsymbol{x}_p(t) &= \boldsymbol{\phi}(\boldsymbol{X}_p, t), \\ \boldsymbol{a}_p(t) &= \boldsymbol{F}_p(t)\boldsymbol{A}_p\boldsymbol{F}_p(t)^T. \end{aligned} \quad (8)$$



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 Eq. (4): 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 [16]. 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.

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 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 the inset figure. We remark that the rotation of view directions is not considered in Wu et al. [45]. Chen et al. [6] 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:

$$f^t(\boldsymbol{d}) = f^0(\boldsymbol{R}^T \boldsymbol{d}). \tag{9}$$

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 [4, 23], the update rule for the world-space covariance matrix $\boldsymbol{a}$ can also be derived by discretizing the rate form of kinematics $\dot{\boldsymbol{a}} = (\nabla \boldsymbol{v})\boldsymbol{a} + \boldsymbol{a}(\nabla \boldsymbol{v})^T$:

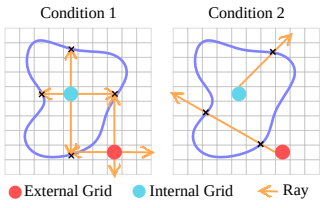
$$\boldsymbol{a}_p^{n+1} = \boldsymbol{a}_i^n + \Delta t(\nabla \boldsymbol{v}_p \boldsymbol{a}_p^n + \boldsymbol{a}_p^n \nabla \boldsymbol{v}_p^T). \tag{10}$$

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 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.

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. [42], we borrow the 3D opacity field from 3D Gaussians

$$d(\boldsymbol{x}) = \textstyle\sum_p \sigma_p \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_p)^T \boldsymbol{A}_p^{-1}(\boldsymbol{x} - \boldsymbol{x}_p)\right). \tag{11}$$



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.

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 $\mathrm{diag}(r_p^2, r_p^2, r_p^2)$, 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.

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:

$$\mathcal{L}_{aniso} = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \max\{\max(\boldsymbol{S}_p)/\min(\boldsymbol{S}_p), r\} - r, \tag{12}$$

where $\boldsymbol{S}_p$ are the scalings of 3D Gaussians [16]. 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.

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.

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 [34], we utilize *fox*, *plane*, and *ruins* from the datasets of Instant-NGP [26], Nerfstudio [41] and the DroneDeploy NeRF [29], 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 using COLMAP [35, 36].

Simulation Setups We build upon the MPM from Zong et al. [53]. 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

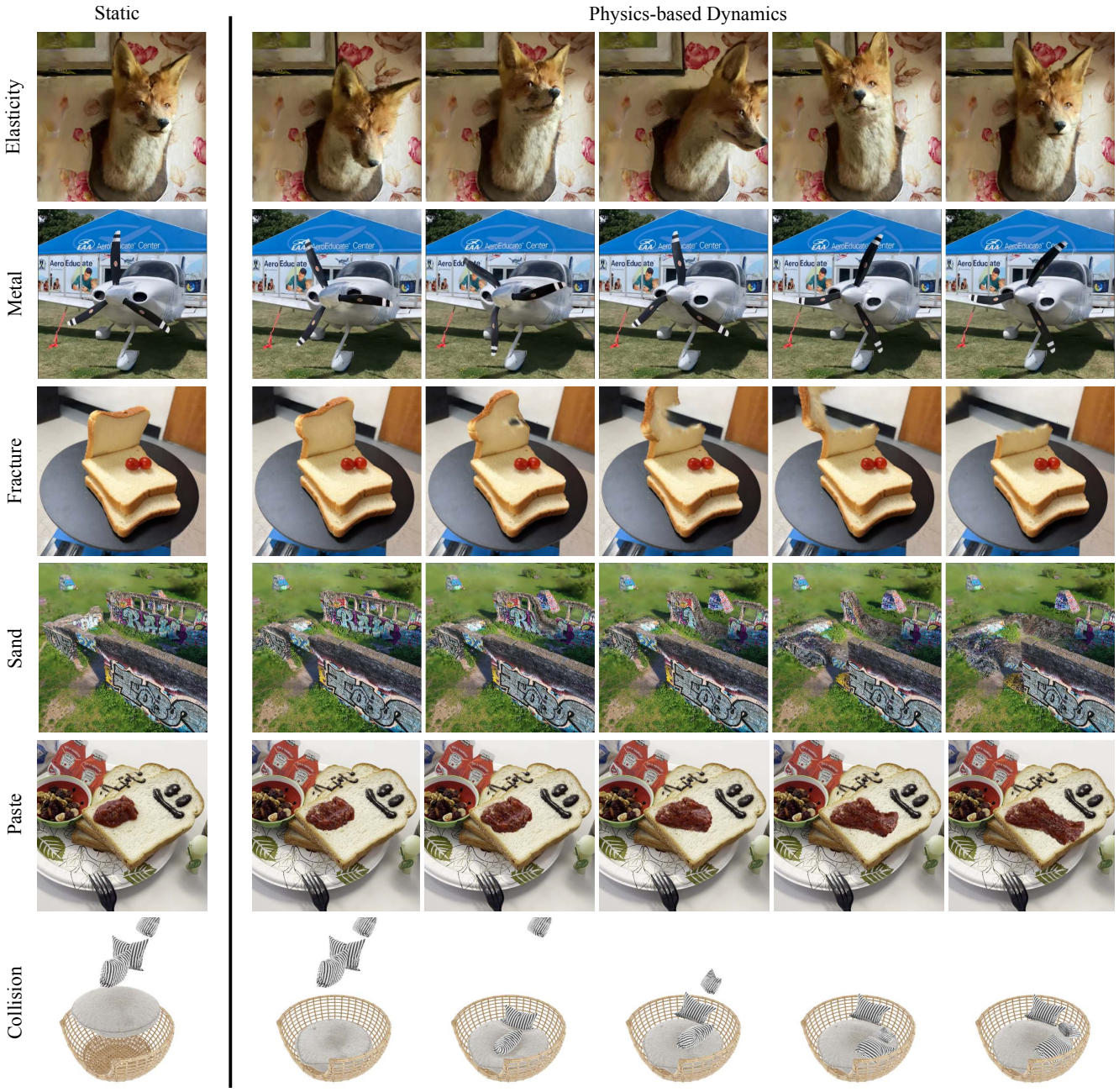


Figure 3. **Material Versatility.** We demonstrate exceptional versatility of our approach across a wide variety of examples: *fox* (elastic entity), *plane* (plastic metal), *toast* (fracture), *ruins* (granular material), *jam* (viscoplastic material), and *sofa suite* (collision).

shown in Fig. 3. 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 [17], which can capture granular-level frictional effects among particles. **Paste** is modeled as viscoplastic non-Newtonian fluid, adhering to Herschel-Bulkley plasticity model [52]. **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

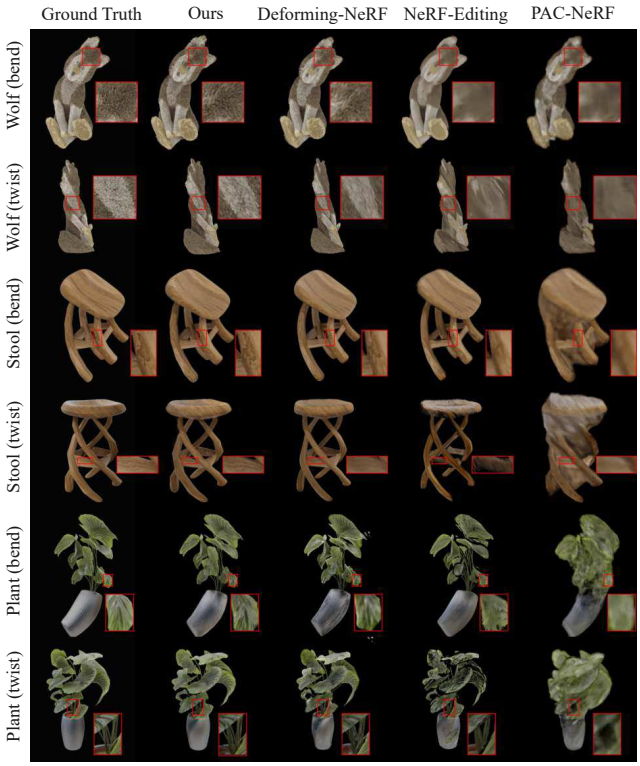


Figure 4. **Comparisons.** For each benchmark case, we select one test viewpoint and visualize all comparisons. We zoom in on some regions to highlight the ability of our method to maintain high-fidelity rendering quality after deformations. We use a black background to avoid the interference of the background.

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.

4.2. Lattice Deformation Benchmarks

Dataset Due to the absence of ground truth for post-deformation, we utilize BlenderNeRF [34] 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 fair comparisons.

Comparisons We compare our method with several state-of-the-art NeRF frameworks that support manual deformations: **1) NeRF-Editing [51]** deforms NeRF using an extracted surface mesh, **2) Deforming-NeRF [47]** utilizes a cage mesh for deformation, and **3) PAC-NeRF [18]** manipulates individual initial particles.

We show qualitative results in Fig. 4 and quantitative results in Tab. 1. NeRF-Editing uses NeuS [43] 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.

Figure 5. **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.

Table 1. 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 [51]	26.74	24.37	25.00	21.10	19.85	19.08
Deforming-NeRF [47]	21.65	21.72	22.32	21.16	17.90	18.63
PAC-NeRF [18]	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**

Ablation Studies We further conduct several ablation studies on these benchmark scenes to validate the necessity of the kinematics of Gaussian kernels and spherical

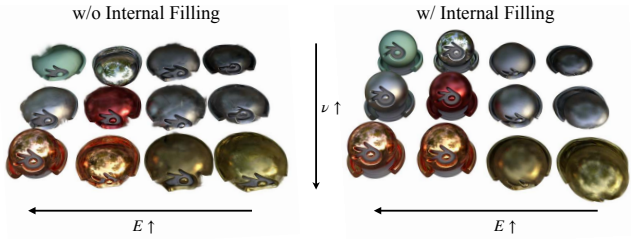


Figure 6. **Internal filling** enables more realistic simulation results. Our method also supports flexible control of dynamics via material parameters. A larger Young's modulus E indicates higher stiffness while a larger poission ratio ν leads to better volume preservation.

harmonics: **1) Fixed Covariance** only translates the Gaussian kernels. **2) Rigid Covariance** only applies rigid transformations on the Gaussians, as assumed in Luiten et al. [22]. **3) Fixed Harmonics** does not rotate the orientations of spherical harmonics, as assumed in Wu et al. [45].

Here we visualize one example in Fig. 5. 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 Tab. 1, which shows that all these enhancements are needed to achieve the best performance of our method.

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. Fig. 6 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, 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 Fig. 7, we conduct a comparison study between our method and NeRF-Editing [51], which utilizes surface As-Rigid-As-Possible (ARAP) deformation [38]. Unlike NeRF-Editing, our approach accurately captures and maintains the volume of the deformed objects.

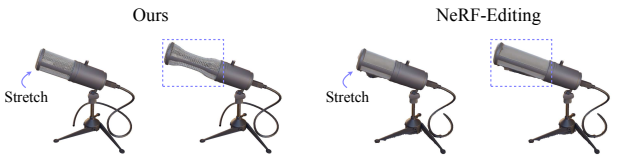


Figure 7. **Volume Conservation.** Compared to the geometry-based editing method [51], our physics-based method is able to capture volumetric behaviors, leading to more realistic dynamics.

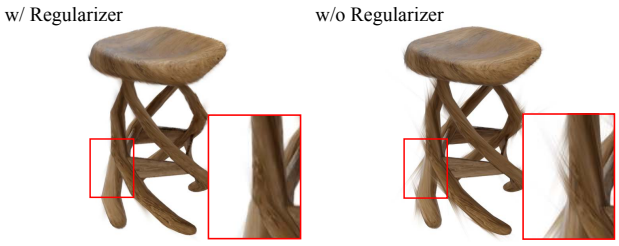


Figure 8. **Anisotropy Regularizer.** We introduce an anisotropy constraint for Gaussian kernels, effectively enhancing the fidelity of the Gaussian-based representation under dynamic conditions.

Anisotropy Regularizer 3D Gaussian models inherently represent anisotropic ellipsoids. However, excessively slender Gaussian kernels can lead to burr-like visual artifacts, especially pronounced during large deformations To tackle this issue, we introduce an additional regularization loss Eq. (12) to constrain anisotropy. As demonstrated in Fig. 8, this additional loss function effectively mitigates the artifacts induced by extreme anisotropy.

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 [3, 50] with a differentiable MPM simulator. Additionally, incorporating geometry-aware 3DGS reconstruction methods [9] 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).

Acknowledgements We thank Ying Nian Wu and Feng Gao for useful discussions. We acknowledge support from NSF (2301040, 2008915, 2244651, 2008564, 2153851, 2023780), UC-MRPI, Sony, Amazon, and TRI.

References

[1] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. 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, 2022. 2

[2] Javier Bonet and Richard D Wood. *Nonlinear continuum mechanics for finite element analysis*. Cambridge university press, 1997. 3, 4

[3] Jiazhong Cen, Jiemin Fang, Chen Yang, Lingxi Xie, Xiaopeng Zhang, Wei Shen, and Qi Tian. Segment any 3d gaussians. *arXiv preprint arXiv:2312.00860*, 2023. 8

[4] S Chandrasekhar. Ellipsoidal figures of equilibrium—an historical account. *Communications on Pure and Applied Mathematics*, 20(2):251–265, 1967. 5

[5] Hsiao-yu Chen, Edith Tretschk, Tuur Stuyck, Petr Kadlecek, Ladislav Kavan, Etienne Vouga, and Christoph Lassner. Virtual elastic objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15827–15837, 2022. 2

[6] Jun-Kun Chen, Jipeng Lyu, and Yu-Xiong Wang. 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, 2023. 2, 4

[7] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022. 2

[8] Ming Gao, Xinlei Wang, Kui Wu, Andre Pradhana, Eftychios Sifakis, Cem Yuksel, and Chenfanfu Jiang. Gpu optimization of material point methods. *ACM Transactions on Graphics (TOG)*, 37(6):1–12, 2018. 3

[9] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. *arXiv preprint arXiv:2311.12775*, 2023. 8

[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. 2

[11] Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics (TOG)*, 38(6): 1–16, 2019. 3

[12] Clément Jambon, Bernhard Kerbl, Georgios Kopanas, Stavros Diolatzis, Thomas Leimkühler, and George" Drettakis. Nerfshop: Interactive editing of neural radiance fields". *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 6(1), 2023. 1

[13] Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. The affine particle-in-cell method. *ACM Transactions on Graphics (TOG)*, 34(4):1–10, 2015. 2

[14] Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. The material point method for simulating continuum materials. In *Acm siggraph 2016 courses*, pages 1–52. 2016. 3, 4

[15] Chenfanfu Jiang, Theodore Gast, and Joseph Teran. Anisotropic elastoplasticity for cloth, knit and hair frictional contact. *ACM Transactions on Graphics (TOG)*, 36(4):1–14, 2017. 2

[16] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (ToG)*, 42(4):1–14, 2023. 1, 2, 3, 4, 5

[17] Gergely Klár, Theodore Gast, Andre Pradhana, Chuyuan Fu, Craig Schroeder, Chenfanfu Jiang, and Joseph Teran. Drucker-prager elastoplasticity for sand animation. *ACM Transactions on Graphics (TOG)*, 35(4):1–12, 2016. 2, 6

[18] 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 *The Eleventh International Conference on Learning Representations*, 2023. 2, 7

[19] Yuan Li, Zhi-Hao Lin, David Forsyth, Jia-Bin Huang, and Shenlong Wang. Climatenerf: Extreme weather synthesis in neural radiance field. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3227–3238, 2023. 2

[20] Zhi-Hao Lin, Wei-Chiu Ma, Hao-Yu Hsu, Yu-Chiang Frank Wang, and Shenlong Wang. 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, 2022. 2

[21] Ruiyang Liu, Jinxu Xiang, Bowen Zhao, Ran Zhang, Jingyi Yu, and Changxi Zheng. Neural impostor: Editing neural radiance fields with explicit shape manipulation. *arXiv preprint arXiv:2310.05391*, 2023. 2

[22] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *arXiv preprint arXiv:2308.09713*, 2023. 2, 8

[23] WILLIAM J McKIVER and David G Dritschel. The motion of a fluid ellipsoid in a general linear background flow. *Journal of Fluid Mechanics*, 474:147–173, 2003. 5

[24] 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. *Communications of the ACM*, 65(1):99–106, 2021. 1, 2, 3

[25] Matthias Müller, Nuttapong Chentanez, and Miles Macklin. Simulating visual geometry. In *Proceedings of the 9th International Conference on Motion in Games*, pages 31–38, 2016. 1

[26] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022. 2, 5

[27] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo

Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5865–5874, 2021. 2

[28] Yicong Peng, Yichao Yan, Shengqi Liu, Yuhao Cheng, Shanyan Guan, Bowen Pan, Guangtao Zhai, and Xiaokang Yang. Cagenerf: Cage-based neural radiance field for generalized 3d deformation and animation. *Advances in Neural Information Processing Systems*, 35:31402–31415, 2022. 1, 2

[29] Nicholas Pilkington. Dronedeploy nerf dataset, 2022. 5

[30] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 2

[31] Yi-Ling Qiao, Alexander Gao, and Ming Lin. Neuphysics: Editable neural geometry and physics from monocular videos. *Advances in Neural Information Processing Systems*, 35:12841–12854, 2022. 2

[32] Yi-Ling Qiao, Alexander Gao, Yiran Xu, Yue Feng, Jia-Bin Huang, and Ming C Lin. Dynamic mesh-aware radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 385–396, 2023. 2

[33] Yuxing Qiu, Samuel Temple Reeve, Minchen Li, Yin Yang, Stuart Ryan Slattery, and Chenfanfu Jiang. A sparse distributed gigascale resolution material point method. *ACM Transactions on Graphics*, 42(2):1–21, 2023. 3

[34] Maxime Raafat. BlenderNeRF, 2023. 5, 7

[35] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5

[36] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 5

[37] Juan C Simo and Thomas JR Hughes. *Computational inelasticity*. Springer Science & Business Media, 2006. 4

[38] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, pages 109–116. Citeseer, 2007. 8

[39] Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. A material point method for snow simulation. *ACM Transactions on Graphics (TOG)*, 32(4): 1–10, 2013. 2, 4

[40] 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. 2

[41] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, et al. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–12, 2023. 5

[42] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653*, 2023. 5

[43] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *Advances in Neural Information Processing Systems*, 2021. 7

[44] Xinlei Wang, Yuxing Qiu, Stuart R Slattery, Yu Fang, Minchen Li, Song-Chun Zhu, Yixin Zhu, Min Tang, Dinesh Manocha, and Chenfanfu Jiang. A massively parallel and scalable multi-gpu material point method. *ACM Transactions on Graphics (TOG)*, 39(4):30–1, 2020. 3

[45] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. *arXiv preprint arXiv:2310.08528*, 2023. 2, 3, 4, 8

[46] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5438–5448, 2022. 2

[47] Tianhan Xu and Tatsuya Harada. Deforming radiance fields with cages. In *European Conference on Computer Vision*, pages 159–175. Springer, 2022. 1, 7

[48] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. *arXiv preprint arXiv:2309.13101*, 2023. 2

[49] Zeyu Yang, Hongye Yang, Zijie Pan, Xiatian Zhu, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. *arXiv preprint arXiv:2310.10642*, 2023. 2

[50] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes. *arXiv preprint arXiv:2312.00732*, 2023. 8

[51] Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, and Lin Gao. Nerf-editing: geometry editing of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18353–18364, 2022. 1, 2, 7, 8

[52] Yonghao Yue, Breannan Smith, Christopher Batty, Changxi Zheng, and Eitan Grinspun. Continuum foam: A material point method for shear-dependent flows. *ACM Transactions on Graphics (TOG)*, 34(5):1–20, 2015. 6

[53] Zeshun Zong, Xuan Li, Minchen Li, Maurizio M Chiaramonte, Wojciech Matusik, Eitan Grinspun, Kevin Carlberg, Chenfanfu Jiang, and Peter Yichen Chen. Neural stress fields for reduced-order elastoplasticity and fracture. *arXiv preprint arXiv:2310.17790*, 2023. 5