Physics3D: Learning Physical Properties of 3D Gaussians via Video Diffusion

Fangfu Liu[1,*] **Hanyang Wang**[1,*] **Shunyu Yao**[2] **Shengjun Zhang**[1]
Jie Zhou[1] **Yueqi Duan**[1,†]
[1]Tsinghua University [2]Stanford University

Abstract

In recent years, there has been rapid development in 3D generation models, opening up new possibilities for applications such as simulating the dynamic movements of 3D objects and customizing their behaviors. However, current 3D generative models tend to focus only on surface features such as color and shape, neglecting the inherent physical properties that govern the behavior of objects in the real world. To accurately simulate physics-aligned dynamics, it is essential to predict the physical properties of materials and incorporate them into the behavior prediction process. Nonetheless, predicting the diverse materials of real-world objects is still challenging due to the complex nature of their physical attributes. In this paper, we propose **Physics3D**, a novel method for learning various physical properties of 3D objects through a video diffusion model. Our approach involves designing a highly generalizable physical simulation system based on a viscoelastic material model, which enables us to simulate a wide range of materials with high-fidelity capabilities. Moreover, we distill the physical priors from a video diffusion model that contains more understanding of realistic object materials. Extensive experiments demonstrate the effectiveness of our method with both elastic and plastic materials. Physics3D shows great potential for bridging the gap between the physical world and virtual neural space, providing a better integration and application of realistic physical principles in virtual environments. Project page: `https://liuff19.github.io/Physics3D`.

1 Introduction

In recent years, 3D computer vision has witnessed significant advancements, with researchers focusing on reconstructing or generating 3D assets [40, 27, 56, 43, 21, 34], and even delving into the realm of 4D dynamics [47, 36]. However, a common feature in these works is the emphasis on color space, which can be difficult in modeling realistic interactive dynamics, especially for applications in areas such as virtual/augmented reality and animation. Physics simulation is one of the most crucial methods to achieve a deeper understanding of the real world and enhance the effectiveness of interactive dynamics. Although conventional methods [53, 12, 28] describe behavior using continuous physical dynamic equations based on body-fixed mesh, they are usually difficult and time-consuming to generate complex 3D objects and suffer from highly nonlinear issues, large deformations or fracture-prone physical phenomena[26].

Powered by recent advances in implicit and explicit 3D representation techniques (*e.g.,* NeRF [40] and 3D Gaussian Splatting [27]), some researchers [65, 35] have attempted to bridge the gap between rendering and simulation using the differentiable Material Point Method (MPM) [22], which enables efficient physical simulation driven by 3D particles (*i.e.,* 3D Gaussian kernels). PhysGaussian [65] extends the capabilities of 3D Gaussian kernels by incorporating physics-based attributes such

[*]Equal contribution. †The corresponding author.

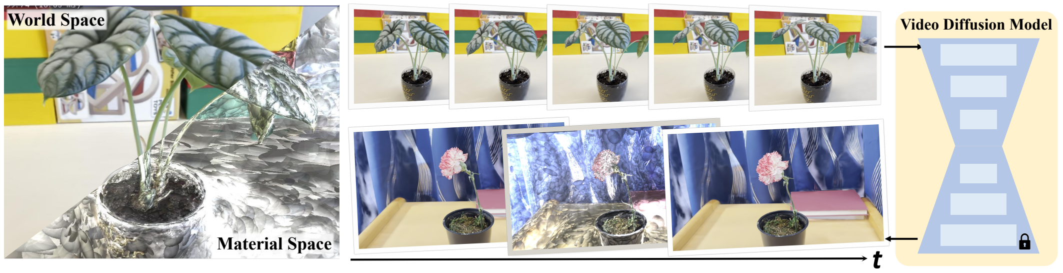

Figure 1: Physics3D is a unified simulation-rendering pipeline based on 3D Gaussians, which learn physics dynamics from video diffusion model.

as velocity, strain, elastic energy, and stress. This unified representation of material substance facilitates both simulation and rendering tasks. However, manual pre-design of physical parameters in PhysGaussian [65] remains a laborious and imprecise process. To avoid manually setting parameters, PhysDreamer [68] leverages object dynamics learned from video generation models [3, 61] to estimate a physical material parameter (*i.e.,* Young's modulus). However, in practical applications, real-world objects often exhibit a complex composite nature, making it challenging for a simulation approach that relies solely on a single physical parameter to fully capture their dynamic behavior. This limits PhysDreamer to be primarily tailored for the simulation of hyper-elastic materials. Specifically, it encounters significant challenges when dealing with materials such as plastics, metals, and non-Newtonian fluids due to its heavy reliance on optimizing Young's modulus alone. The inherent complexities in these materials surpass the capabilities of PhysDreamer, highlighting the need for a more comprehensive and robust approach that considers a broader range of physical properties for accurate and effective simulation.

In this paper, we propose **Physics3D**, a generalizable physical simulation system to learn various physical properties of 3D objects. Given a 3D Gaussian representation, we first expand the dimension of the physical parameters to capture both elasticity and viscosity. Then we design a viscoelastic Material Point Method (MPM) to simulate 3D dynamics. Through the simulation process, we decompose the deformation gradient into two separate components and calculate them independently to contribute to the overall force. Finally, leveraging the capabilities of the differentiable MPM, we optimize the physical parameters via the Score Distillation Sampling (SDS) [43] strategy to distill physical priors from the video diffusion model. Iterating the MPM process and SDS optimization, Physics3D achieves high-fidelity and realistic performance in a wide range of materials. Extensive experiments demonstrate the efficacy and superiority of our proposed Physics3D over existing methods. In summary, our key contributions are as follows.

- We propose a novel generalizable physical simulation system called Physics3D, which is capable of learning diverse material 3D dynamics in a physically plausible manner.
- We introduce a viscoelastic Material Point Method to simulate both the viscosity and elasticity of 3D dynamics, enabling support of various material simulations.
- We design a learnable internal filling strategy to optimize part of 3D Gaussians and utilize the Score Distillation Sampling to optimize physical parameters from the video diffusion model.
- Experiments show Physics3D is effective in creating high-fidelity and realistic 3D dynamics, ready for various interactions across users and objects in the future.

2 Related Work

Dynamic 3D representations. Rapid advancements in static 3D representations have sparked interest in incorporating temporal dynamics into the 3D modeling of dynamic objects and scenes. Various explicit or hybrid representation techniques have demonstrated impressive outcomes, including planar decomposition for 4D space-time grids [8, 51, 13], the utilization of NeRF representation [35, 44, 14], and alternative structural approaches [59, 1, 10]. Recently, 3D Gaussian Splatting [27] has revolutionized the representation via its outstanding rendering efficiency and high-quality results. Efforts have been made to extend static 3D Gaussians into dynamic versions, yielding promising results. Dynamic 3D Gaussians [39] refine per-frame Gaussian Splatting through dynamic regularizations and shared properties such as size, color, and opacity. Similarly, the concept of 4D Gaussian Splatting [64, 66] employs a deformation network to anticipate time-dependent positional, scaling, and rotational deformations. In addition, DreamGaussian4D [47] learns motion from image-conditioned generated videos [4], enabling more controllable and diverse 3D motion representations.

Viscoelastic materials. In the realm of computer graphics and animation, there has been significant interest in accurately simulating the behavior of nonrigid objects and their interactions with physical environments. Conventional elastic models [58, 69] predicated on Hooke's law [49] are the cornerstone for simulating the deformation of objects. These models are effective in representing materials that exhibit perfectly elastic behavior, returning to their original shape after the applied force is removed. However, real-world materials often exhibit more complex behaviors that cannot be captured by simple elastic models. The introduction of viscoelastic materials in computer graphics [57] has expanded the range of simulated material behaviors. Viscoelastic materials combine the characteristics of both viscous fluids and elastic solids, leading to time-dependent deformations under constant stress, a phenomenon known as creep [9]. Compared with elastic models, viscoelastic models offer a more versatile framework for animating nonrigid objects. They can simulate the slow restoration of a material's shape after the cessation of force, as well as the permanent deformation that occurs due to prolonged stress.

Video generation models. With the emergence of models like Sora [7], the field of video generation [60, 63, 2, 5] has drawn significant attention. These powerful video models [32, 52, 18, 20] are typically trained on extensive datasets of high-quality video content. Sora [7] is capable of producing minute-long videos with realistic motions and consistent viewpoints. Furthermore, some large-scale video models, such as Sora [7], can even support physically plausible effects. Inspired by these capabilities, we aim to distill the physical principles observed in videos and apply them to our static 3D objects, thereby achieving more realistic and physically accurate results. In our framework, we choose Stable Video Diffusion [3] to optimize our physical properties.

3 Problem Formulation

Given a static representation through 3D Gaussians, our goal is to estimate the physical attributes of each Gaussian particle and generate physics-plausible motions by organizing the interaction of force and velocity among these particles. These physical properties include mass (m), Young's modulus (E), and Poisson's ratio (ν). Young's modulus and Poisson's ratio controls dynamics of elastic objects. For example, with a fix amount of external force applied, system will higher Young's modulus will have smaller deformation.

However, only modeling the property of elastic objects is inadequate for recovering diverse physics with heterogeneous materials in real-world applications, which significantly limits the recent work like PhysDreamer [68]. For example, they usually suffer from complex mixed materials, especially in scenarios of rapid deformation where viscosity emerges as a significant factor in dynamics. Therefore, our key insight is to build a more comprehensive physics model that includes additional parameters, notably viscosity, to enrich the descriptive capacity for real-world objects especially in inelastic scenarios. With a closer look at the viscosity coefficient ν_v and ν_d, it serves as a viscous damper to exert resistance against rapid deformation, thus enhancing the model's fidelity in capturing viscoelasticity.

To model viscoelastic stresses with physical fidelity, we explore continuum mechanics, where Lamé constants (also referred to as Lamé coefficients or parameters), denoted by λ and μ, emerge as pertinent material-related quantities within the strain-stress relationship, while viscosity coefficient ν_v and ν_d governs the viscous dynamics. Conventionally, λ and μ are respectively denoted as Lamé's first and second parameters. Their significance varies in different contexts. For example, in fluid dynamics, μ is related to the dynamic viscosity of fluids while in elastic environments, Lamé parameters λ and μ intertwine with Young's modulus (E) and Poisson's ratio (ν) via a specific relationship. On the other hand, viscosity coefficient correct the elastic strain-stress relation by dynamically controlling the relation to account for viscosity. Consequently, our framework pivots towards the estimation of the viscosity coefficient (ν_v and ν_d) and the two Lamé parameters, λ, and μ. As for other physical attributes, we align with the conventional methods [68, 65], where the particle mass (m_p) is pre-calculated as the product of a constant density (ρ) and the particle volume (V_p), and Poisson's ratio is constant across the object.

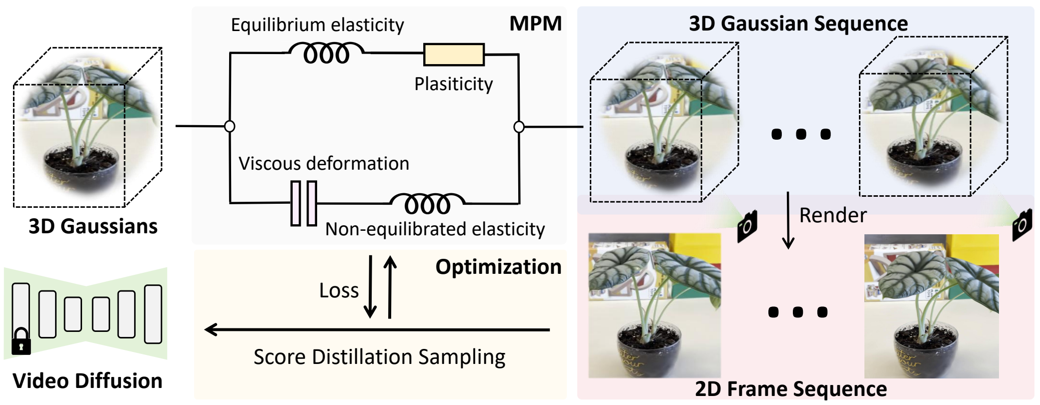

Figure 2: **Pipeline of Physics3D**. Given an object represented as 3D Gaussians, We first simulate it using the Material Point Method (MPM). The simulation comprises two distinct components: an elastoplastic part and a viscoelastic part. These components operate independently to calculate the individual stresses within the object and are combined to determine the overall stress. After the simulation, a series of Gaussians with varying orientations are generated, reflecting the dynamic evolution of the scene. Then, we render these Gaussians from a fixed viewpoint to produce a sequence of video frames. Finally, we utilize a pretrained video diffusion model with Score Distillation Sampling (SDS) strategy to iteratively optimize physical parameters.

4 Method

In this section, we introduce our method, *i.e.*, Physics3D, for learning the dynamics of multi-material 3D systems with physical alignment. Our goal is to estimate the various physical properties of 3D objects. Building upon this, we first review the theory of three foundational techniques (referenced in Sec. 4.1) that form the backbone of our algorithm. Then we introduce our physical property estimation framework (Sec. 4.2) and describe our particle-based simulation process. Finally, we present the optimization process (Sec. 4.3) to further improve the accuracy and efficiency. An overview of our framework is depicted in Figure 2.

4.1 Preliminary

Continuum Mechanics describes motions by a deformation map $\mathbf{x} = \phi(\mathbf{X}, t)$ from the material space Ω^0 (with coordinate $\mathbf{X}$) to the world space Ω^n (with coordinate $\mathbf{x}$). The deformation gradient $\mathbf{F} = \frac{\partial \phi}{\partial \mathbf{X}}(\mathbf{X}, t)$ measures local rotation and strain [6]. We consider viscoelastic materials where we have two components [17], the elastoplastic component $\mathbf{F_E F_P}$ and the viscoelastic component $\mathbf{F_N F_V}$. They are in parallel combination shown in Figure 3, and formulated as:

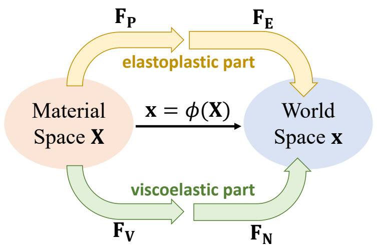

Figure 3: Elastoplastic and viscoelastic decomposition.

$$\mathbf{F} = \mathbf{F_E F_P} = \mathbf{F_N F_V}. \tag{1}$$

In this work, we take $\mathbf{F_P} = \mathbf{I}$ for convenience. Intuitively, we model our material with two different compounds in parallel connection as they deform in the same way thus having the same total strain. However, only the elastic components $\mathbf{F_E}$ and $\mathbf{F_N}$ contributes to the internal stress $\boldsymbol{\sigma_E}$ and $\boldsymbol{\sigma_N}$.

We can now evolve the system with dynamical equations. Denoting the velocity field with $\mathbf{v}(\mathbf{x}, t)$ and density field with $\rho(\mathbf{x}, t)$, the conservation of momentum and conservation of mass [15] is given by:

$$\rho \frac{D\mathbf{v}}{Dt} = \nabla \cdot \boldsymbol{\sigma} + \mathbf{f}, \quad \frac{D\rho}{Dt} + \rho \nabla \cdot \mathbf{v} = 0, \tag{2}$$

where $\mathbf{f}$ denotes an external force, $\boldsymbol{\sigma} = \boldsymbol{\sigma_E} + \boldsymbol{\sigma_N}$ is the total internal stress. We will also need to update the strain tensor after updating the material point, and we refer to the Appendix for more details.

Material Point Method (MPM) [55, 26, 22] discretizes the material into deformable particles and employs particles to monitor the complete history of strain and stress states while relying on a background grid for precisely evaluating derivatives during force computations. This methodology has demonstrated its efficacy in simulating diverse materials [25, 31, 54] and is proved to be capable of simulating some viscoelastic and viscoplastic materials [46, 67]. MPM operates in a particle-to-grid (P2G) and grid-to-particle (G2P) transfer loop. In P2G process, MPM transfers mass and momentum from particles to grids:

$$m_i^n = \sum_p w_{ip}^n m_p, \quad m_i^n \mathbf{v_i^n} = \sum_p w_{ip}^n m_p (\mathbf{v_p^n} + C_p^n(\mathbf{x_i} - \mathbf{x_p^n})), \tag{3}$$

Here i and p represent the fields on the Eulerian grid and the Lagrangian particles respectively. Each particle p carries a set of properties including volume V_p, mass m_p, position $\mathbf{x_p^n}$, velocity $\mathbf{v_p^n}$, deformation gradient $\mathbf{F_p^n}$ and affine momentum C_p^n at time t_n. w_{ip}^n is the B-spline kernel defined on i-th grid evaluated at $\mathbf{x_p^n}$. After P2G, the transferred grids can be updated as:

$$\mathbf{v_i^{n+1}} = \mathbf{v_i^n} - \frac{\Delta t}{m_i} \sum_p \tau_p^n \nabla w_{ip}^n V_p^0 + \Delta t \mathbf{g}, \tag{4}$$

here $\mathbf{g}$ represents the acceleration due to gravity. Then G2P transfers velocities back to particles and updates particle states τ (*i.e.,* Kirchhoff tensor).

$$\tau_p^{n+1} = \boldsymbol{\tau}(\mathbf{F_E^{n+1}}, \mathbf{F_N^{n+1}}), \tag{5}$$

where $\mathbf{F_E^{n+1}}$ and $\mathbf{F_E^{n+1}}$ are two parts of strain tensor. We will provide a detailed introduction in Sec. 4.2. In this work, we integrate the physical properties of viscoelastic materials into the Material Point Method, thereby enhancing the generalization capabilities of MPM. This implementation enables the simulation for a wide range of materials commonly found in the real world, including various inelastic materials.

Score Distillation Sampling (SDS) is first introduced by DreamFusion [43], which distills the 3D knowledge from large 2D pretrain model [50]. For a set of particles parameterized by physical properties θ, its rendering $\mathbf{x}$ can be obtained by $\mathbf{x} = g(\theta)$ where g is a differentiable renderer. SDS calculates the gradients of physical parameters θ by,

$$\nabla_\theta \mathcal{L}_{\text{SDS}}(\phi, \mathbf{x} = \mathbf{g}(\theta)) = \mathbb{E}_{t,\boldsymbol{\epsilon}} \left[w(t)(\boldsymbol{\epsilon}_\phi(\mathbf{x}_t; y, t) - \boldsymbol{\epsilon}) \frac{\partial \mathbf{x}_t}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \theta} \right], \tag{6}$$

where $w(t)$ is a weighting function that depends on the timestep t and y denotes the given condition. $\epsilon_\theta(\mathbf{x}_t, t)$ is autoencoder in diffusion model to estimate the origin distribution of the given noise $\mathbf{x}_t$.

4.2 Physical Properties with Viscoelasticity Deformation

To model physical properties, we employ MLS-MPM [23] as our simulator, and we formalize the simulation process for a single sub-step as follows:

$$\mathbf{x^{n+1}}, \mathbf{v^{n+1}}, \mathbf{F^{n+1}}, C^{n+1} = \mathcal{S}(\mathbf{x^n}, \mathbf{v^n}, \mathbf{F^n}, C^n, \theta, \Delta t), \tag{7}$$

Here, θ contains the physical properties of all particles: mass m_i, Young's modulus E_i, Poisson's ratio ν_i, Lamé coefficients λ, μ, viscosity coefficient ν_{Ni} and volume V_i. Δt denotes the simulation step size. Within the MPM simulation, stress, as depicted in Equation 1, can be divided into two components: one representing elastoplasticity, denoted as $\mathbf{F_E}$, and the other referring to viscoelasticity, denoted as $\mathbf{F_N}$. The two parts are also illustrated in Figure 2, consisting of a parallel combination: (a) an elastic spring with a frictional element for plasticity and; (b) another spring with a viscous dash-pot element which is assumed to capture the dissipation. Now, we elaborate on the computation for each component individually.

Model for elastoplastic part. We first explain how to compute the internal stress $\boldsymbol{\sigma_E}$ through $\mathbf{F_E}$, which is essential in updating kinematical variables like velocities v through Eq. 2, for the elastic branch. We will speak out the rule here but postpone explanations and intuitions in Appendix B.3. As we demonstrate in Sec. 4.1 and Appendix B.2, one can compute the Cauchy stress tensor given the energy function. In this work, we choose the fixed corotated constitutive model for the elastic part, whose energy function is

$$\psi(\mathbf{F_E}) = \psi(\boldsymbol{\Sigma_E}) = \mu_E \sum_i (\sigma_{E,i} - 1)^2 + \frac{\lambda_E}{2}(\det(\mathbf{F_E}) - 1)^2, \tag{8}$$

where $\boldsymbol{\Sigma_E}$ is the diagonal singular value matrix $\mathbf{F_E} = \mathbf{U}\boldsymbol{\Sigma_E}\mathbf{V^T}$. And $\sigma_{E,i}$ are singular values of $\mathbf{F_E}$. From this energy function, we can compute the Cauchy stress tensor as:

$$\boldsymbol{\sigma_E} = \frac{2\mu}{\det(\mathbf{F_E})}\left(\mathbf{F_E} - \mathbf{R}\right)\mathbf{F_E^T} + \lambda_E(\det(\mathbf{F_E}) - 1), \tag{9}$$

where $\mathbf{R} = \mathbf{UV^T}$. We now explain how to update the $\mathbf{F_E^n}$ with the velocity field $\mathbf{v^n}$ at nth step. For purely elastic case, this can be simply done via:

$$\mathbf{F_E^{n+1}} = (\mathbf{I} + \Delta t \nabla \mathbf{v^n})\mathbf{F^n}, \tag{10}$$

where Δt is the length of the time segment in the MPM method. This formula represents the fact that the internal velocity field causes a change in strains.

Model for viscoelastic part. We now explain the other branch of our model: the viscoelastic part. The total strain now consist of two parts $\mathbf{F} = \mathbf{F_N}\mathbf{F_V}$. There are two key features of this brunch. First, only the $\mathbf{F_N}$ contributes to the internal stress. Secondly, $\mathbf{F_V}$ only plays a role in the update rule of $\mathbf{F_N}$. For the first step, we know have a relation between internal stress $\boldsymbol{\sigma_N}$ (or equivalently Kirchoff tensor $\boldsymbol{\tau_N} = \det(\mathbf{F_N})\boldsymbol{\sigma_N}$):

$$\tau_N = 2\mu_N \epsilon_N + \lambda_N \text{tr}(\epsilon_N)\mathbf{I} \tag{11}$$

where we denote τ_N as a vector of singular value of $\boldsymbol{\tau_N}$, in another word $\boldsymbol{\tau_N} = U\text{diag}(\tau_N)V^T$. And ϵ_N, called log principle Kirchoff tensor, denotes a vector takes the diagonal element of $\log \Sigma_N$, where Σ_N as the diagonal singular value matrix $\mathbf{F_N} = U\Sigma_N V^T$. We can again update kinemetical variables afterward. The second step is we need to modify the update rule for $\mathbf{F_N}$ tensor. This boils down to a trial-and-correction procedure where we first update

$$\mathbf{F^n_{N,\text{tr}}} = (\mathbf{I} + \Delta t \nabla \mathbf{v^n})\mathbf{F^n_N}, \tag{12}$$

then we modify the trial strain tensor $\mathbf{F^n_{N,\text{tr}}}$ by

$$\epsilon_N^{n+1} = A(\epsilon^n_{N,\text{tr}} - B\text{tr}(\epsilon^n_{N,\text{tr}}) \cdot \mathbf{1}) \tag{13}$$

where ϵ_N^{n+1} denotes the log principle Kirchoff tensor of $\mathbf{F_N^{n+1}}$. A and B are functions of viscocity parameters ν_d and ν_e. In this work, we take $\nu_d = \nu_e = \nu_N$ for simplicity. Please refer to Appendix B.3 for more detailed intuitions and explanations.

4.3 Optimization with Video Diffusion Model

Through iterations of the Material Point Method (MPM), we obtain a set of Gaussians evolving over time t. To ensure consistency, the camera remains stationary as we capture and render each Gaussian to produce a reference image I_t^r. To supervise and optimize this process, we employ two optional models: image-to-video and text-to-video diffusion models. For video diffusion model, we utilize Stable Video Diffusion [3] that learns rich 2D diffusion prior from large real-world data. Then, we use SDS loss to guide our optimization process as:

$$\nabla_\theta \mathcal{L}_{\text{SDS}} = \mathbb{E}_{t,p,\epsilon}\left[w(t)(\epsilon_\phi(I_t^p; t, I_t^r, \Delta p, y) - \epsilon)\frac{\partial I_t^p}{\partial \theta}\right]. \tag{14}$$

Here, $w(t)$ represents a time-dependent weighting function, $\epsilon_\phi(\cdot)$ denotes the predicted noise generated by the 2D diffusion prior ϕ, Δp signifies the relative change in camera pose from the reference camera r, and y denotes the given condition (*i.e.,* image or text). Furthermore, we adopt a partial filling strategy like [65], where internal volumes of select solid objects are optionally filled to augment simulation realisticity. For better rendering quality, we optimize the filled Gaussian along with the SDS process. The optimization objective for filled Gaussians is defined as:

$$\mathcal{L}_{\text{Fill}} = \frac{1}{T}\sum_{t=1}^{T} \lambda ||I_t^p - I_t^r||_2^2 \tag{15}$$

The two losses mentioned above are optimized separately.

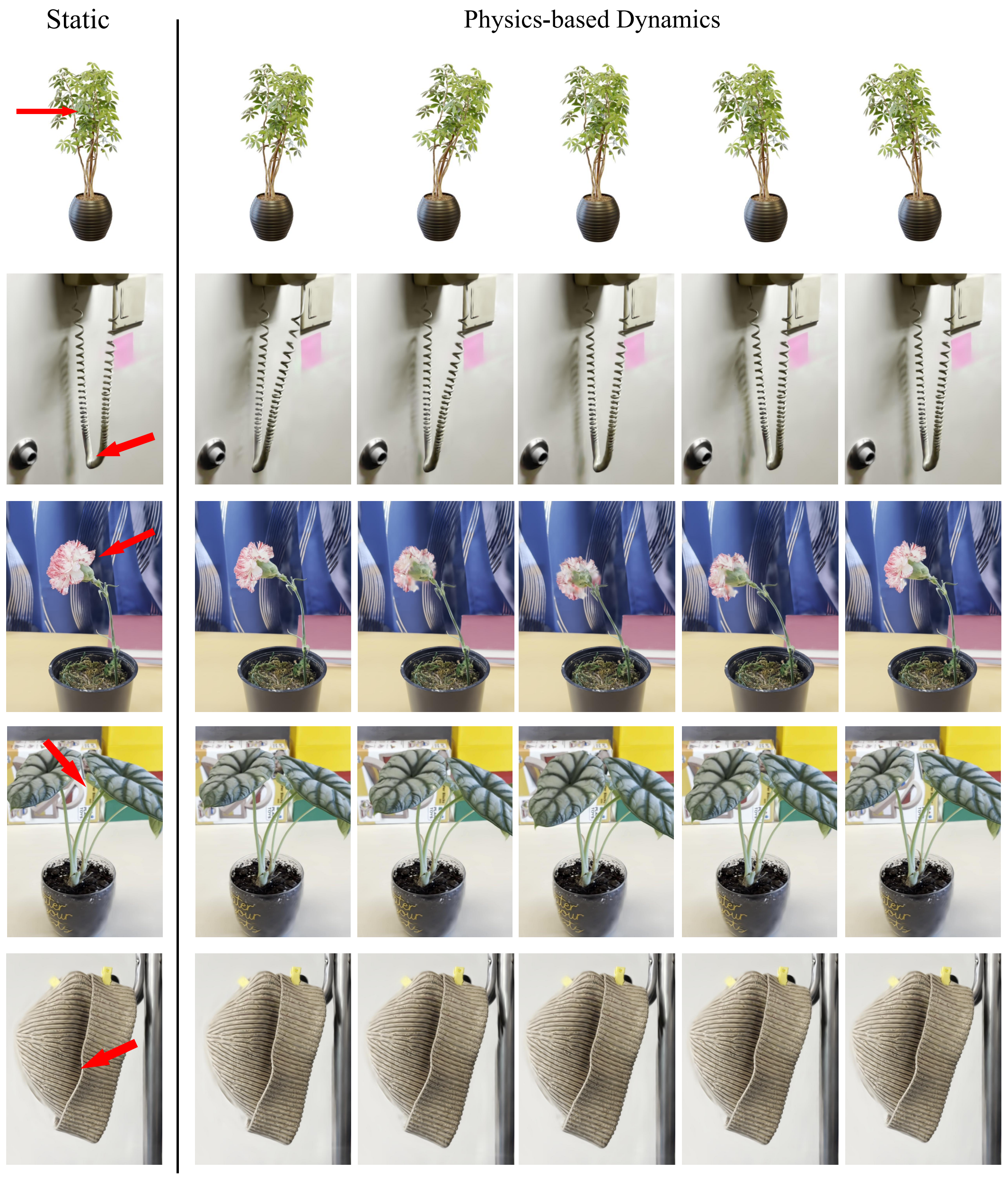

Figure 4: **Visual results of Physics3D** on different subjects with an external force (red arrow). Physics3D is able to generate realistic scene movement while maintaining good motion consistency.

5 Experiments

In this section, we conduct extensive experiments to evaluate our Physics3D and show the comparison results against other methods [65, 68, 47]. We first present our qualitative results and comparisons with baselines (Sec. 5.2). Then we report the quantitative results with a user study (Sec. 5.3). Finally, we carry out more open settings and ablation studies to further verify the efficacy of our framework design (Sec. 5.4). Please refer to the Appendix for more visualizations and detailed analysis.

5.1 Experiment Setup

Datasets. We evaluate our method for generating diverse dynamics using several sources of input. We choose four real-world static scenes from PhysDreamer [68] for fair comparison. Each scene includes an object and a background. The objects include a carnation, an alocasia plant, a telephone

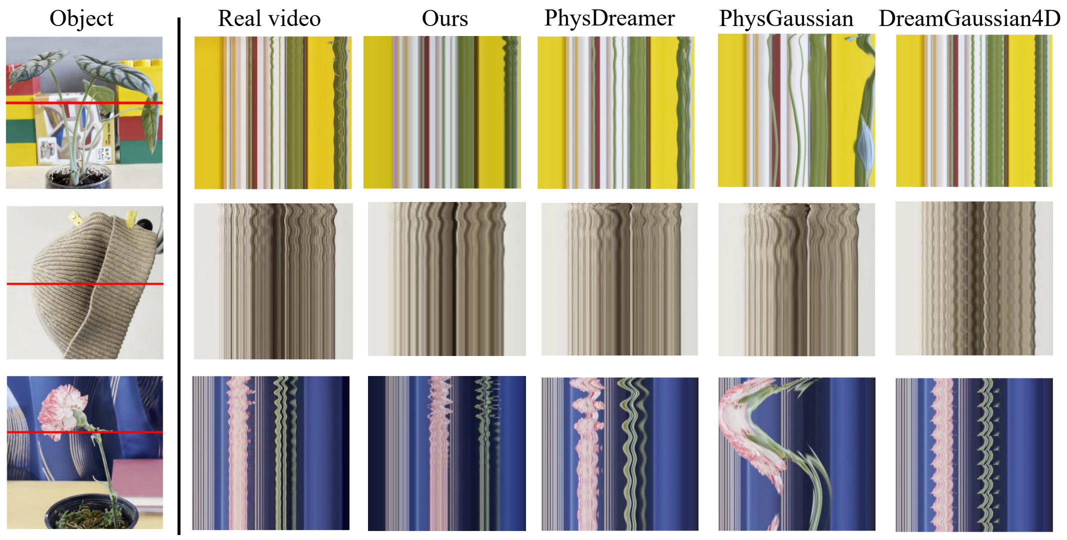

Figure 5: **Comparison results** on different subjects. From the visual results, we observe that PhysDreamer [68] only estimates the elastic properties of objects, resulting in the lack of damping. DreamGaussian4D [47] and PhysGaussian [65] are respectively limited in unrealistically constant, low-magnitude periodic motion and low-frequency movements. In contrast, our model successfully balances high and low-frequency oscillations with more realistic damping.

cord, and a beanie hat. We also employed NeRF datasets [40] to evaluate the efficacy of our proposed model. Additionally, we utilize BlenderNeRF [45] to synthesize several scenes.

Implementation Details. In our implementation, we initiate the process by reconstructing 3D Gaussians from multi-view images, establishing a foundational representation of the scene. In complex realistic cases, we undertake a segmentation step to differentiate between the background and foreground elements, focusing solely on the latter for subsequent simulation tasks. Prior to simulation, we execute internal particle filling operations to refine the representation further. Each Gaussian kernel is then associated with a distinct set of physical properties targeted for optimization, facilitating the fine-tuning process. Subsequently, we discretize the foreground region into a grid structure, typically set at dimensions of 50^3. As for the MPM simulation, we employ 400 sub-steps within each temporal interval spanning successive video frames. This temporal granularity translates to a sub-step duration of $1e-4$ second, ensuring precision and accuracy in the simulation dynamics. Notably, the optimization process for each object requires approximately 5 minutes to complete on a single NVIDIA A6000 (48GB) GPU.

Baselines and Metrics. We extensively compare our method with three baselines: PhysDreamer [68], PhysGaussian [65] and DreamGaussian4D [47]. Following [68], we show our results with notable comparisons with space-time slices. For metrics, we evaluate our approach with the video-quality metrics: PSNR, SSIM, MS-SSIM, and VMAF on the rendered videos. We also conduct a user study to further demonstrate the overall quality and the alignment with real-world physics in our simulation, which can be found in our Appendix.

5.2 Qualitative Results

Figure 4 shows qualitative results of simulated interactive motion. For each case, we visualize one example with its initial scene and deformation sequence. The results demonstrate our model's capability of simulating the movement of complex textured objects, presenting a realistic and physically plausible outcome. Additional experiments are included in the Appendix A.1.

Following PhysDreamer [68], we compare our results with real captured videos and simulations from other methods [68, 65, 47] in Figure 5. We utilize space-time slices to present our comparisons. These slices depict time along the vertical axis and spatial slices of the object along the horizontal axis, as indicated by the red lines in the "object" column. Through these visualizations, we aim to elucidate the magnitude and frequencies of the oscillating motions under scrutiny. PhysDreamer [68] closely approximates the elastic properties of objects, resulting in periodic oscillations with subtle damping, contrasting the unrealistic aspects. PhysGaussian [65] showcases unrealistically low-frequency movements due to inaccurate parameter settings. DreamGaussian4D [47] lacks physical prior and generates unrealistically constant, low-magnitude periodic motion. In contrast, our model successfully balances high and low-frequency oscillations with more realistic damping, aligning more closely with the behavior of objects in the real world.

5.3 Quantitative Results

Table 1 shows the average video-quality metrics over rendered videos from the same fixed perspective. Results clearly demonstrate higher scores for Physics3D, indicating better video quality and motion consistency of our results. We also conduct numerous user experiments in Appendix A.3.

Table 1: **Quantitative comparisons** on rendered videos using different video-quality metrics.

	PSNR↑	SSIM↑	MS-SSIM↑	VMAF↑
PhysDreamer [68]	13.89	0.55	0.37	0.52
PhysGaussian [65]	13.86	0.57	0.39	0.59
Ours	**14.72**	**0.59**	**0.49**	**0.59**
- *w/o* viscoelastic part	14.13	0.53	0.36	0.52
- *w/o* elastoplastic part	13.53	0.55	0.41	0.50

5.4 Ablation Study

We conduct ablation study in Figure 6 to evaluate the efficacy of our physics modeling process. Specifically, we investigate the importance of elastoplastic and viscoelastic components from the model architecture. We observe that the removal of either of these modules leads to a notable degradation in the realism of the physical simulations. Particularly, the absence of the elastoplastic component results in a lack of elasticity, making objects more susceptible to shape deformation and fluid-like behavior. On the other hand, the absence of the viscoelastic component leads to a deficiency in sustained damping and rebound effects, especially in scenarios with minimal external disturbances where energy dissipation occurs rapidly. These show the significance of both components of our model in capturing the dynamics of physical objects. Please refer to our Appendix for more ablations.

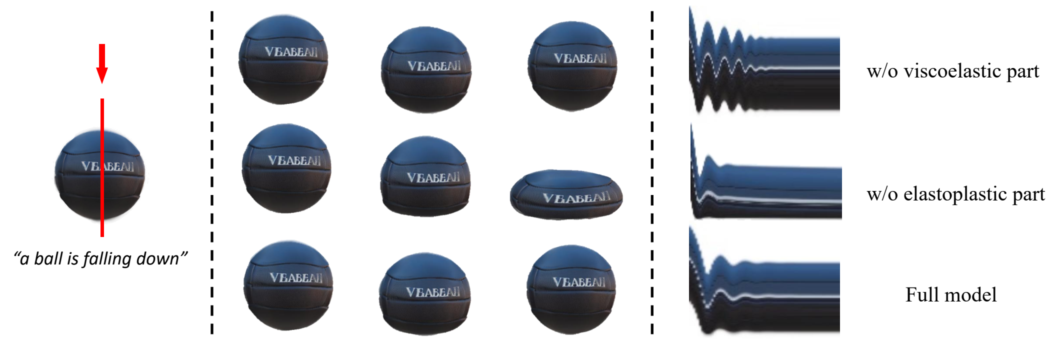

Figure 6: Ablation study on elastoplastic and viscoelastic parts of our model.

6 Conclusion

In this paper, we present Physics3D, a novel framework to learn various physical properties of 3D objects from video diffusion model. Our method tackles the challenge of estimating diverse material properties by incorporating two key components: elastoplastic and viscoelastic modules. The elastoplastic component facilitates simulations of pure elasticity, while the viscoelastic module introduces damping effects, crucial for capturing the behavior of materials exhibiting both elasticity and viscosity. Furthermore, we leverage a video generation model to distill inherent physical priors, improving our understanding of realistic material characteristics. Extensive experiments show Physics3D is effective in creating high-fidelity and realistic 3D dynamics.

Limitations and Future Work. In complex environments with a lot of entangled objects, our method requires manual intervention to assign the scope of movable objects and define the filling ranges for objects, which is not efficient for more real applications. In the future, we aim to utilize the prior of large segmentation models to solve the problem with more comprehensive physics system modeling. We believe that Physics3D takes a significant step to open up a wide range of applications from realistic simulations to interactive virtual experiences and will inspire more works in the future.

References

[1] Jad Abou-Chakra, Feras Dayoub, and Niko Sünderhauf. Particlenerf: A particle-based encoding for online neural radiance fields. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5975–5984, 2024.

[2] Omer Bar-Tal, Hila Chefer, Omer Tov, Charles Herrmann, Roni Paiss, Shiran Zada, Ariel Ephrat, Junhwa Hur, Yuanzhen Li, Tomer Michaeli, et al. Lumiere: A space-time diffusion model for video generation. *arXiv preprint arXiv:2401.12945*, 2024.

[3] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

[4] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

[5] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22563–22575, 2023.

[6] Javier Bonet and Richard D Wood. *Nonlinear continuum mechanics for finite element analysis*. Cambridge University Press, 1997.

[7] Tim Brooks, Bill Peebles, Connor Homes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. 2024.

[8] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 130–141, 2023.

[9] Richard M Christensen. *Theory of viscoelasticity*. Courier Corporation, 2003.

[10] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022.

[11] Yu Fang, Minchen Li, Ming Gao, and Chenfanfu Jiang. Silly rubber: an implicit material point method for simulating non-equilibrated viscoelastic and elastoplastic solids. *ACM Transactions on Graphics (TOG)*, 38(4):1–13, 2019.

[12] Carlos A Felippa. Introduction to finite element methods. *University of Colorado*, 885, 2004.

[13] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12479–12488, 2023.

[14] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5712–5721, 2021.

[15] Paul Germain. Functional concepts in continuum mechanics. *Meccanica*, 33:433–444, 1998.

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[17] Sanjay Govindjee and Stefanie Reese. A presentation and comparison of two large deformation viscoelasticity models. 1997.

[18] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

[19] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[20] Wenyi Hong, Ming Ding, Wendi Zheng, Xinghan Liu, and Jie Tang. Cogvideo: Large-scale pretraining for text-to-video generation via transformers. *arXiv preprint arXiv:2205.15868*, 2022.

[21] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. Lrm: Large reconstruction model for single image to 3d. In *The Twelfth International Conference on Learning Representations*, 2023.

[22] 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.

[23] 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.

[24] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

[25] 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.

[26] 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.

[27] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023.

[28] Axel Kilian and John Ochsendorf. Particle-spring systems for structural form finding. *Journal of the international association for shell and spatial structures*, 46(2):77–84, 2005.

[29] Diederik P Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *NeurIPS*, 2021.

[30] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[31] 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.

[32] Dan Kondratyuk, Lijun Yu, Xiuye Gu, José Lezama, Jonathan Huang, Rachel Hornung, Hartwig Adam, Hassan Akbari, Yair Alon, Vighnesh Birodkar, et al. Videopoet: A large language model for zero-shot video generation. *arXiv preprint arXiv:2312.14125*, 2023.

[33] Agelos Kratimenos, Jiahui Lei, and Kostas Daniilidis. Dynmf: Neural motion factorization for real-time dynamic view synthesis with 3d gaussian splatting. *arXiv preprint arXiv:2312.00112*, 2023.

[34] Xiaoyu Li, Qi Zhang, Di Kang, Weihao Cheng, Yiming Gao, Jingbo Zhang, Zhihao Liang, Jing Liao, Yan-Pei Cao, and Ying Shan. Advances in 3d generation: A survey. *arXiv preprint arXiv:2401.17807*, 2024.

[35] 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*, 2022.

[36] Huan Ling, Seung Wook Kim, Antonio Torralba, Sanja Fidler, and Karsten Kreis. Align your gaussians: Text-to-4d with dynamic 3d gaussians and composed diffusion models. *arXiv preprint arXiv:2312.13763*, 2023.

[37] Fangfu Liu, Hanyang Wang, Weiliang Chen, Haowen Sun, and Yueqi Duan. Make-Your-3D: Fast and consistent subject-driven 3d content generation. *arXiv preprint arXiv:2403.09625*, 2024.

[38] Fangfu Liu, Diankun Wu, Yi Wei, Yongming Rao, and Yueqi Duan. Sherpa3D: Boosting high-fidelity text-to-3d generation via coarse 3d prior. *arXiv preprint arXiv:2312.06655*, 2023.

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

[40] 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.

[41] Richard A Newcombe, Dieter Fox, and Steven M Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[42] Alexander Quinn Nichol and Prafulla Dhariwal. Improved Denoising Diffusion Probabilistic Models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

[43] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *The Eleventh International Conference on Learning Representations*, 2022.

[44] 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.

[45] Maxime Raafat. BlenderNeRF, May 2023.

[46] Daniel Ram, Theodore Gast, Chenfanfu Jiang, Craig Schroeder, Alexey Stomakhin, Joseph Teran, and Pirouz Kavehpour. A material point method for viscoelastic fluids, foams and sponges. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 157–163, 2015.

[47] Jiawei Ren, Liang Pan, Jiaxiang Tang, Chi Zhang, Ang Cao, Gang Zeng, and Ziwei Liu. Dreamgaussian4d: Generative 4d gaussian splatting. *arXiv preprint arXiv:2312.17142*, 2023.

[48] Herbert E. Robbins. *An Empirical Bayes Approach to Statistics*. Springer New York, 1992.

[49] Jan Rychlewski. On hooke's law. *Journal of Applied Mathematics and Mechanics*, 48(3):303–314, 1984.

[50] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022.

[51] Ruizhi Shao, Zerong Zheng, Hanzhang Tu, Boning Liu, Hongwen Zhang, and Yebin Liu. Tensor4d: Efficient neural 4d decomposition for high-fidelity dynamic reconstruction and rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16632–16642, 2023.

[52] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, et al. Make-a-video: Text-to-video generation without text-video data. *arXiv preprint arXiv:2209.14792*, 2022.

[53] David E Stewart. Rigid-body dynamics with friction and impact. *SIAM review*, 42(1):3–39, 2000.

[54] 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.

[55] Deborah Sulsky, Shi-Jian Zhou, and Howard L Schreyer. Application of a particle-in-cell method to solid mechanics. *Computer physics communications*, 87(1-2):236–252, 1995.

[56] 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.

[57] Demetri Terzopoulos and Kurt Fleischer. Modeling inelastic deformation: viscolelasticity, plasticity, fracture. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 269–278, 1988.

[58] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 205–214, 1987.

[59] Haithem Turki, Jason Y Zhang, Francesco Ferroni, and Deva Ramanan. Suds: Scalable urban dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12375–12385, 2023.

[60] Ruben Villegas, Mohammad Babaeizadeh, Pieter-Jan Kindermans, Hernan Moraldo, Han Zhang, Mohammad Taghi Saffar, Santiago Castro, Julius Kunze, and Dumitru Erhan. Phenaki: Variable length video generation from open domain textual descriptions. In *International Conference on Learning Representations*, 2022.

[61] Jiuniu Wang, Hangjie Yuan, Dayou Chen, Yingya Zhang, Xiang Wang, and Shiwei Zhang. Modelscope text-to-video technical report. *arXiv preprint arXiv:2308.06571*, 2023.

[62] Zhengyi Wang, Yikai Wang, Yifei Chen, Chendong Xiang, Shuo Chen, Dajiang Yu, Chongxuan Li, Hang Su, and Jun Zhu. Crm: Single image to 3d textured mesh with convolutional reconstruction model. *arXiv preprint arXiv:2403.05034*, 2024.

[63] Chenfei Wu, Jian Liang, Lei Ji, Fan Yang, Yuejian Fang, Daxin Jiang, and Nan Duan. Nüwa: Visual synthesis pre-training for neural visual world creation. In *European conference on computer vision*, pages 720–736. Springer, 2022.

[64] 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.

[65] Tianyi Xie, Zeshun Zong, Yuxin Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. *arXiv preprint arXiv:2311.12198*, 2023.

[66] 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.

[67] 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.

[68] Tianyuan Zhang, Hong-Xing Yu, Rundi Wu, Brandon Y Feng, Changxi Zheng, Noah Snavely, Jiajun Wu, and William T Freeman. Physdreamer: Physics-based interaction with 3d objects via video generation. *arXiv preprint arXiv:2404.13026*, 2024.

[69] 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. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–11, 2023.

Appendix

A More Results

To further demonstrate the effectiveness and impressive visualization results of our Physics3D, we conducted more experiments including additional visual results and user study.

A.1 Additional Visual Results

Figure 7 shows additional visual results of interactive video sequences. We use BlenderNeRF [45] to synthesize the cases below. For each case, We apply external forces of different directions and magnitudes to static objects and render video frames to show their motion states.

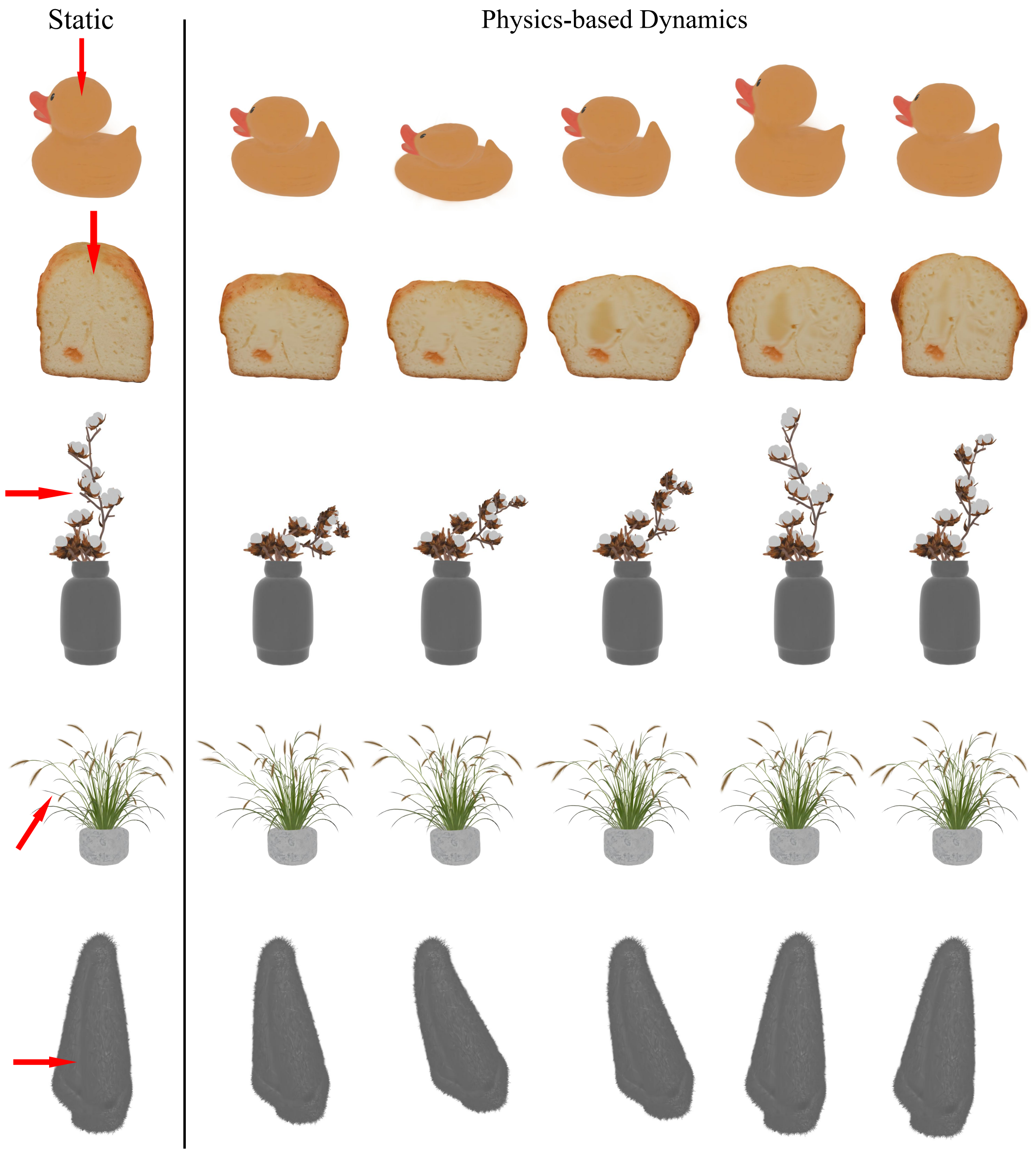

Figure 7: More visual results on different objects. Notice ours can simulate a variety of non-elastic and composite materials, including *rubber*, *fluffy bread*, *fabric* etc.

A.2 Additional Ablation Study

We carry out additional ablation analyses on the Physics3D design in Figure 6 to assess the efficacy of our physics modeling process. Specifically, we investigate the impact of removing the elastoplastic and viscoelastic components from the model architecture. The findings highlight that the removal of either of these modules leads to a notable decrease in the realisticity of the physical simulations. Particularly, the absence of the elastoplastic component results in a lack of elasticity, thereby making objects more susceptible to shape deformation and fluid-like behavior. On the other hand, the absence of the viscoelastic component leads to a deficiency in sustained damping and rebound effects, especially in scenarios with minimal external disturbances where energy dissipation occurs rapidly. These outcomes underscore the significance of both components of our model in capturing the dynamics of physical objects. Notice that without internal filling, the hollow part of the cake is prone to collapse in the simulation process, but with internal filling, the object's deformation under external force is more realistic and smooth.

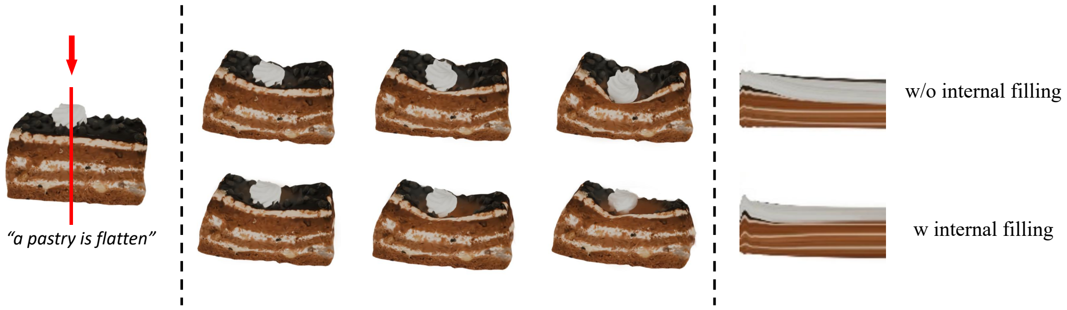

Figure 8: Ablation study on internal filling.

A.3 User Study

For user study, we show each volunteer with five samples of rendered video from a random method (PhysDreamer [68], PhysGaussian [65], DreamGaussian4D [47], and ours) and real-captured videos. The study engaged 30 volunteers to assess the generated results in 20 rounds. In each round, they were asked to select the video they preferred the most, based on quality, realisticity, alignment with input 3D object, and fluency. We find our method is significantly preferred by users over these aspects.

Table 2: **Quantitative comparison results** of PhysGaussian [65], PhysDreamer [68] and our Physics3D on the action coherence, motion realism, and overall quality score in a user study, rated on a range of 1-10, with higher scores indicating better performance.

Method	Action Coherence	Motion Realism	Overall Quality
PhysGaussian [65]	7.82	6.89	6.93
PhysDreamer [68]	8.76	7.73	7.89
Physics3D (Ours)	**8.95**	**8.57**	**9.05**

B More Analysis

B.1 Material Point Method (MPM) Algorithm

The Material Point Method (MPM) simulates the behavior of materials by discretizing a continuum body into particles and updating their properties over time. Here's an additional summary of the MPM algorithm:

Particle to Grid Transfer. Mass and momentum are transferred from particles to grid nodes. This step involves distributing the particle properties (mass and velocity) to nearby grid points.

$$
\begin{aligned}
m_i^n &= \sum_p w_{ip}^n m_p, \\
m_i^n \mathbf{v}_i^n &= \sum_p w_{ip}^n m_p (\mathbf{v_p^n} + \mathbf{C}_p^n(\mathbf{x_i} - \mathbf{x_p^n})).
\end{aligned}
$$

Grid Update. Grid velocities are updated based on external forces and the forces from neighboring particles. This step moves the grid points according to the applied forces.

$$
\mathbf{v_i^{n+1}} = \mathbf{v_i^n} - \frac{\Delta t}{m_i} \sum_p \boldsymbol{\tau}_p^n \nabla w_{ip}^n V_p^0 + \Delta t \mathbf{g}.
$$

Grid to Particle Transfer. Velocities are transferred back to particles, and particle states are updated. This step brings the changes in grid velocities back to the particles.

$$
\begin{aligned}
\mathbf{v_p^{n+1}} &= \sum_i \mathbf{v_i^{n+1}} w_{ip}^n, \\
\mathbf{x_p^{n+1}} &= \mathbf{x_p^n} + \Delta t \mathbf{v_p^{n+1}}, \\
\boldsymbol{C}_p^{n+1} &= \frac{12}{\Delta x^2 (b+1)} \sum_i w_{ip}^n \mathbf{v_i^{n+1}} \left(\mathbf{x_i^n} - \mathbf{x_p^n}\right)^T, \\
\nabla \mathbf{v_p^{n+1}} &= \sum_i \boldsymbol{v}_i^{n+1} \nabla w_{ip}^{n\,T}, \\
\tau_p^{n+1} &= \boldsymbol{\tau}(\mathbf{F_E^{n+1}}, \mathbf{F_N^{n+1}}),
\end{aligned}
$$

Here b is the B-spline degree, and Δx is the Eulerian grid spacing. We extensively demonstrate how to update $\mathbf{F_E}$, $\mathbf{F_N}$ and τ in Appendix B.2.

B.2 Dynamic 3D Generation.

Dynamic 3D generation aims to synthesize the dynamic behavior of 3D objects or scenes in the process of generating three-dimensional representations. Unlike static 3D generation methods [38, 21, 62, 37] that solely focus on the spatial morphology of objects, the field of 3D dynamics imposes a higher requirement, necessitating the incorporation of information from all three spatial dimensions as well as the temporal dimension. As advancements in 3D representation techniques continue to emerge, certain parameterizable 3D representations have empowered us to inject dynamic information into 3D models [35, 33]. One popular approach [41, 47] integrates three essential components:

Gaussian Splatting [27] represents 3D information with a set of 3D Gaussian kernels. Each Gaussian can be described with a center $x \in \mathbb{R}^3$, a scaling factor $s \in \mathbb{R}^3$, and a rotation quaternion $q \in \mathbb{R}^4$. Additionally, an opacity value $\alpha \in \mathbb{R}$ and a color feature $c \in \mathbb{R}^3$ are for volumetric rendering. These parameters can be collectively denoted by θ, with $\theta_i = \{x_i, s_i, q_i, \alpha_i, c_i\}$ representing the parameters for the i-th Gaussian. The volume rendering color C of each pixel is computed by blending N ordered points overlapping the pixel:

$$
C = \sum_{i \in N} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j). \tag{16}
$$

Diffusion Model [19, 42] is usually pre-trained on large 2D datasets to provide a foundational motion prior for dynamic 3D generation. These models, characterized by their probabilistic nature, are tailored to acquire knowledge of the data distribution $p(x_0)$ through a step-by-step denoising process applied to a normally distributed variable. Throughout the training phase, the data distribution is perturbed towards an isotropic Gaussian distribution over T timesteps, guided by a predefined noising schedule $\alpha_t \in (0, 1)$, where $\overline{\alpha_t} = \sum_{s=1}^{t} \alpha_s$ and t uniformly sampled from $1, ..., T$:

$$
\mathbf{z}_t = \sqrt{\bar{\alpha}_t} \mathbf{x_0} + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}, \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{17}
$$

The backward denoising process estimates the origin distribution by autoencoder $\epsilon_\theta(\mathbf{z}_t, t)$. The final training loss can be simplified as:

$$\mathcal{L}_{DM} = \mathbb{E}_{\mathbf{x},\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\phi\left(\mathbf{z}_t, t\right)\|_2^2\right]. \tag{18}$$

Score Distillation Sampling (SDS) is a technique used in machine learning, particularly in the context of generative models, to improve sample quality and diversity. It addresses the challenge of generating high-quality samples from complex probability distributions, such as those learned by deep neural networks.

In traditional generative models like Generative Adversarial Networks (GANs) [16] or Variational Autoencoders (VAEs) [30], generating samples involves directly sampling from the learned latent space. However, this approach often results in low-quality samples with poor diversity, especially in regions of low probability density.

SDS introduces a novel sampling strategy that leverages the notion of score matching. Score matching, introduced by Hyvärinen in 2005 [24], is a technique for training generative models by matching the score function (gradient of the log-density) of the model distribution to that of the true data distribution.

Given a target distribution $p_{\text{data}}(\mathbf{x})$ and a model distribution $p_\phi(\mathbf{x})$, where ϕ represents the parameters of the model, the score function is defined as:

$$\nabla_{\mathbf{x}} \log p_\phi(\mathbf{x}) \tag{19}$$

The score function provides valuable information about the local geometry of the probability distribution, helping to guide the sampling process towards regions of high probability density.

In SDS, the goal is to distill the score function learned by a complex generative model into a simpler, more tractable form. This distilled score function can then be used to guide the sampling process efficiently, resulting in higher-quality samples with improved diversity.

Formally, given a set of generated samples $\{\mathbf{x}_i\}_{i=1}^N$ from the model distribution $p_\phi(\mathbf{x})$, the distilled score function $\hat{s}(\mathbf{x})$ is learned to approximate the score function of the model distribution. This is achieved by minimizing the score matching loss:

$$\mathcal{L}_{\text{SM}}(\hat{s}) = \frac{1}{N}\sum_{i=1}^{N} \|\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}_i) - \hat{s}(\mathbf{x}_i)\|^2 \tag{20}$$

where $\|\cdot\|$ denotes some norm (e.g., L^2 norm) and $\hat{s}(\mathbf{x}_i)$ is the estimated score at sample $\mathbf{x}_i$.

As for particular diffusion model, its score function can be related to the predicted noise (shown in Eq. 18) for the smoothed density through Tweedie's formula [48]:

$$\boldsymbol{\epsilon}_\phi\left(\mathbf{z}_t, t\right) = -\sigma_t s_\phi(\mathbf{z}_t; t). \tag{21}$$

Training the diffusion model with a (weighted) evidence lower bound (ELBO) simplifies to a weighted denoising score matching objective for parameters ϕ [19, 29]:

$$\mathcal{L}_{\text{Diff}}(\phi, \mathbf{x}) = \mathbb{E}_{t\sim\mathcal{U}(0,1),\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[w(t)\|\epsilon_\phi(\alpha_t\mathbf{x} + \sigma_t\epsilon; t) - \epsilon\|_2^2\right], \tag{22}$$

where $w(t)$ is a weighting function that depends on the timestep t. To understand the difficulties of this approach, consider the gradient of $\mathcal{L}_{\text{Diff}}$:

$$\nabla_\theta \mathcal{L}_{\text{Diff}}(\phi, \mathbf{x} = g(\theta)) = \mathbb{E}_{t,\epsilon}\left[w(t)\underbrace{\left(\hat{\epsilon}_\phi(\mathbf{z}_t; y, t) - \epsilon\right)}_{\text{Noise Residual}}\underbrace{\frac{\partial\hat{\epsilon}_\phi(\mathbf{z}_t; y, t)}{\partial\mathbf{z}_t}}_{\text{U-Net Jacobian}}\underbrace{\frac{\partial\mathbf{x}}{\partial\theta}}_{\text{Generator Jacobian}}\right] \tag{23}$$

where following DreamFusion [43], we absorb the constant $\alpha_t\mathbf{I} = \partial\mathbf{z}_t/\partial x$ into $w(t)$. In practice, the U-Net Jacobian term is expensive to compute (requires backpropagating through the diffusion model U-Net), and poorly conditioned for small noise levels as it is trained to approximate the scaled Hessian of the marginal density. In [43], It is found that omitting the U-Net Jacobian term leads to an effective gradient:

$$\nabla_\theta \mathcal{L}_{\text{SDS}}(\phi, \mathbf{x} = g(\theta)) \triangleq \mathbb{E}_{t,\epsilon}\left[w(t)\left(\hat{\epsilon}_\phi(\mathbf{z}_t; y, t) - \epsilon\right)\frac{\partial\mathbf{x}}{\partial\theta}\right], \tag{24}$$

here, we get the gradient of a weighted probability density distillation loss in Eq. 6.

B.3 Additional Analysis in Continuum Mechanics

Basic intuitions in continuum mechanics involves explaining the strain tensor $\mathbf{F}$ which describes the internal deformation of the material, and $\boldsymbol{\sigma}$ which describe the internal stress tensor. In an actual material, there might be multiple compounds, each of them might have their own strain and stress tensor and they might interact with each other. For purpose of this subsection, we explain the intuition when there is only one such compounds. However we will generalize it into multiple compounds case later in the paper.

One could describe an equilibrium material as a point cloud $\mathbf{X}$. In actually simulation, one take a discretization of the system. For our convenience, we will just state the intuition in continum, and generalization to discrete points should be straightforward. When the material is away from its equilibrium position, the position of point is given by $\mathbf{x} \neq \mathbf{X}$. However, we notice that if, for example, we move $\mathbf{x}$ and the adjacent point $\tilde{\mathbf{x}}$ in the same way, then there is actually no internal deformation of the material. In another word, internal deformation describe how relative position between two points $\mathbf{x}$ and its adjacent $\tilde{\mathbf{x}}$ change. To make this quantity well-defined, we normalize it with the original difference between $\mathbf{X}$ and $\tilde{\mathbf{X}}$

$$\tilde{\mathbf{F}}_{ij} = \frac{\tilde{\mathbf{x}}_{\mathbf{i}} - \mathbf{x}_i}{\tilde{\mathbf{X}}_{\mathbf{j}} - \mathbf{X}_j} \tag{25}$$

Taking continuum limit, the equation becomes differential and reproduce

$$\mathbf{F} = \frac{\partial \phi}{\partial \mathbf{X}}(\mathbf{X}, t) \tag{26}$$

in the main text.

To get the dynamics, we will further need the stress tensor $\boldsymbol{\sigma}$. However, the general relation between $\boldsymbol{\sigma}$ and $\mathbf{F}$ is usually complicated. In fact only when internal stress is a conservative force, we can related it to $\mathbf{F}$ through an energy function $\psi(\mathbf{F})$. An analog for this case is the Hooke's force in a spring, where we can introduce a potential energy for the force. There are two main types of internal stress people use in the literature. The first Piola-Kirchoff stress

$$\mathbf{P} = \frac{\partial \psi}{\partial \mathbf{F}} \tag{27}$$

and a related Cauchy stress

$$\boldsymbol{\sigma} = \frac{1}{\det(\mathbf{F})} \frac{\partial \psi}{\partial \mathbf{F}} \mathbf{F}^T \tag{28}$$

We will use Cauchy stress for most part of out paper. However, we could sometimes use the other one interchangeably. Physically, if we want the force acting on a unit surface with normal vector $\hat{n}$, the force will be $\boldsymbol{\sigma} \cdot \hat{n}$. Another version people use is the Kirchoff stress $\boldsymbol{\tau} = \det(\mathbf{F})\boldsymbol{\sigma}$. The only difference between them are whether it is measured with the deformed volume($\boldsymbol{\sigma}$) or the undeformed volume($\boldsymbol{\tau}$). Sometimes using one or the other could be easier for technical reason, as we can see later in the viscoelastic model.

Beyond this simple case, the relation will be quite complicated and non-universal. In this paper, we discuss a specific variant of them, which we explain below. One can see both the relation between $\boldsymbol{\sigma}$ and $\mathbf{F}$, and the update rule for $\mathbf{F}$ get modified.

With these background, we can explain the intuition behind Eq.2. The second equation

$$\frac{D\rho}{Dt} + \rho \nabla \cdot \mathbf{v} = 0, \tag{29}$$

is nothing but mass conservation, $\rho \nabla \cdot \mathbf{v}$ is the local density times the local divergence, which physically correspond to the mass flows out of a local unit volumn in unit time.

The first equation is Newton's law

$$\rho \frac{D\mathbf{v}}{Dt} = \nabla \cdot \boldsymbol{\sigma} + \mathbf{f}, \tag{30}$$

the $\mathbf{f}$ term is the external force. The term $\nabla \cdot \boldsymbol{\sigma}$ is the divergence for the stress tensor, thus giving the total force on a unit volume material.

Model for elastoplastic part. We explain how to compute the internal stress $\boldsymbol{\sigma_E}$ through $\mathbf{F_E}$, and how to update $\mathbf{F_E}$ for the elastic part in this appendix. Again for current paper we consider the elastoplastic branch with only elastic part. As we have reviewed, one can compute the Cauchy stress tensor given the energy function. In this work, we choose the Fixed corotated constitutive model for the elastic part, whose energy function is

$$\psi(\mathbf{F_E}) = \psi(\boldsymbol{\Sigma_E}) = \mu \sum_i (\sigma_{E,i} - 1)^2 + \frac{\lambda}{2}(\det(\mathbf{F_E}) - 1)^2 \tag{31}$$

in which $\boldsymbol{\Sigma_E}$ is the diagonal singular value matrix $\mathbf{F_E} = \mathbf{U}\boldsymbol{\Sigma_E}\mathbf{V^T}$. And $\sigma_{E,i}$s are singular values of $\mathbf{F_E}$. From this energy function, one can compute the Cauchy stress tensor

$$\boldsymbol{\sigma_E} = \frac{2\mu}{\det(\mathbf{F_E})}\left(\mathbf{F_E} - \mathbf{R}\right)\mathbf{F_E^T} + \lambda(\det(\mathbf{F_E}) - 1) \tag{32}$$

in which $\mathbf{R} = \mathbf{U}\mathbf{V^T}$.

We now explain how to update the $\mathbf{F_E^n}$ with the velocity field $\mathbf{v^n}$ at nth step. For purely elastic case, this can be simply done via

$$\mathbf{F_E^{n+1}} = (\mathbf{I} + \Delta t \nabla \mathbf{v^n})\mathbf{F^n} \tag{33}$$

in which Δt is the length of the time segment in the MPM method. This formula represents the fact that internal velocity field cause change in strains. However, as one will say for the viscoelastic part, this rule will change.

Model for viscoelastic part. We now explain the other branch of our model: the viscoelastic part. Now we should imagine a elastic strain series connect with a viscous dissipator. In another word, the total strain $\mathbf{F} = \mathbf{F_N}\mathbf{F_V}$. We now explain two key features for this brunch. First, only the $\mathbf{F_N}$ contributes to the internal stress. Secondly, $\mathbf{F_V}$ only plays a role in the update rule of $\mathbf{F_N}$. Very roughly, one can view the dissipator $\mathbf{F_V}$ as a reservoir for strain and each time during the update, people will be able to relocate part of the strain into the $\mathbf{F_V}$, such that only part of the strain contributes to the internal stress.

For the $\mathbf{F_N}$ part, we again take it to be a elastic system, so the relation between strain and stress is again given by the energy function. Here we choose the energy function following [11]

$$\psi_N(\Sigma_N) = \mu_N \text{tr}((\log \Sigma_N)^2) + \frac{1}{2}\lambda_N(\text{tr}(\log \Sigma_N))^2 \tag{34}$$

In this formula, we introduce Σ_N as the diagonal singular value matrix $\mathbf{F_N} = U\Sigma_N V^T$. As one can see, this potential is only a function of the singular values, so the Kirchoff tensor $\boldsymbol{\tau_N}$ will be diagonal in the same basis as the strain $\mathbf{F_N}$. So we will just write down the relation between $\mathbf{F_N}$ and $\boldsymbol{\tau_N}$ just in the singular value basis

$$\tau_N = \frac{\partial \psi_N(\Sigma_N)}{\partial \epsilon_N} = 2\mu_N \epsilon_N + \lambda_N \text{tr}(\epsilon_N)\mathbf{I} \tag{35}$$

where we denote τ_N as a vector of singular value of $\boldsymbol{\tau_N}$, in another word $\boldsymbol{\tau_N} = U\text{diag}(\tau_N)V^T$. This is sometimes called principle Kirchoff tensor in literature. And ϵ_N, called log principle Kirchoff tensor, denotes a vector takes the diagonal element of $\log \Sigma_N$. So this relation is essentially a vector equation. We can recover the Kirchoff tensor $\boldsymbol{\tau_N}$ thus the Cauchy stress tensor

$$\boldsymbol{\sigma_N} = \frac{1}{\det(\mathbf{F_N})}\boldsymbol{\tau_N}. \tag{36}$$

Now we can discuss the update rule for the strain tensor $\mathbf{F_N}$. As we explained before, the rough intuition was part of the strain tensor $\mathbf{F_N}$ can dissipate into the dissipator $\mathbf{F_V}$. More quantitatively, we can follow the dissipator model in [11], which results into a trial-and-correction procedure. The idea is we first imagine $\mathbf{F_N}$ evolve elastically

$$\mathbf{F^n_{N,\text{tr}}} = (\mathbf{I} + \Delta t \nabla \mathbf{v^n})\mathbf{F_N^n} \tag{37}$$

Next step, we modify this trial strain tensor by introducing a dissipation into the dissipator

$$\epsilon_N^{n+1} = \epsilon^n_{N,\text{tr}} - \Delta t \frac{\partial \psi_V}{\partial \tau_N} \tag{38}$$

Here again, we assume the dissipation happen to only the singular value of the strain tensor, so $\epsilon_{N,\text{tr}}^{n}$ is the log principle Kirchoff tensor of $\mathbf{F}_{\mathbf{N},\text{tr}}^{\mathbf{n}}$ and ψ_V is the dissipation potential. We take a model where

$$\psi_V\left(\tau_N\right)=\frac{1}{2\nu_d}\left|\operatorname{dev}\left(\tau_N\right)\right|^2+\frac{1}{9\nu_v}\left(\tau_N\cdot\mathbf{1}\right)^2 \tag{39}$$

where ν_d and ν_v are parameters controlling the dissipation of the deviatoric and dilational parts. In this work, we take them to be equal, representing the homogeneity and isotropicity of the material. This is merely for simplifying the simulation and can be released straightforwardly. We will not use them directly instead, we can put this equation back into Eq.38, and get a version of this formula purely in terms of ϵ_N^{n+1} and $\epsilon_{N,\text{tr}}^{n}$

$$\epsilon_N^{n+1}=A(\epsilon_{N,\text{tr}}^{n}-B\text{tr}(\epsilon_{N,\text{tr}}^{n})\cdot\mathbf{1}) \tag{40}$$

In principle, we can write the parameters A, B in terms of ν_d, ν_v, however since in our algorithm, all these parameters will be learnt from videos, we will directly learn parameter A and B in the update rule without bothering ν_d, ν_v. But it worth remembering our model comes from a dissipation potential.

C Ethical Statement

We confirm that all data used in this paper for research and publication have been obtained and used in a manner compliant with ethical standards. The individuals engaged in all experiments have given consent for their use, or the data are sourced from publicly available datasets and were used in accordance with the terms of use and permissions. Furthermore, the publication and use of these data and models do not pose any societal or ethical harm. We have taken necessary precautions to ensure that the research presented in this paper respects individual rights, including the right to privacy and the fundamental principles of ethical research conduct.