Reconstruction and Simulation of Elastic Objects with Spring-Mass 3D Gaussians

Licheng Zhong[1*], Hong-Xing Yu[1], Jiajun Wu[1], and Yunzhu Li[1,2,3]

[1] Stanford University, Stanford, USA
[2] Columbia University, New York, USA
[3] University of Illinois Urbana-Champaign, Champaign, USA

Abstract. Reconstructing and simulating elastic objects from visual observations is crucial for applications in computer vision and robotics. Existing methods, such as 3D Gaussians, model 3D appearance and geometry, but lack the ability to estimate physical properties for objects and simulate them. The core challenge lies in integrating an expressive yet efficient physical dynamics model. We propose Spring-Gaus, a 3D physical object representation for reconstructing and simulating elastic objects from videos of the object from multiple viewpoints. In particular, we develop and integrate a 3D Spring-Mass model into 3D Gaussian kernels, enabling the reconstruction of the visual appearance, shape, and physical dynamics of the object. Our approach enables future prediction and simulation under various initial states and environmental properties. We evaluate Spring-Gaus on both synthetic and real-world datasets, demonstrating accurate reconstruction and simulation of elastic objects. Project page: https://zlicheng.com/spring_gaus.

Keywords: System Identification · Spring-Mass Model · 3D Gaussians

1 Introduction

Reconstructing and simulating elastic objects from visual observations poses a fundamental challenge in computer vision and robotics, with applications spanning virtual reality, augmented reality, and robotic manipulation. Accurately modeling the elasticity of objects is crucial for creating immersive experiences and enabling embodied agents to understand and interact with the elastic objects commonly encountered in our daily lives. However, accurately identifying the dynamics from vision still presents considerable challenges.

Existing methods for dynamic scene reconstruction, such as 3D Gaussians and their dynamic extensions [14, 26, 44, 49], have made significant progress in capturing the temporal changes of the appearance and geometry of objects. However, these methods do not capture the physical properties of the reconstructed

[*] The work was done while L. Zhong was a visiting student at Stanford University. L. Zhong is now at Shanghai Jiao Tong University.

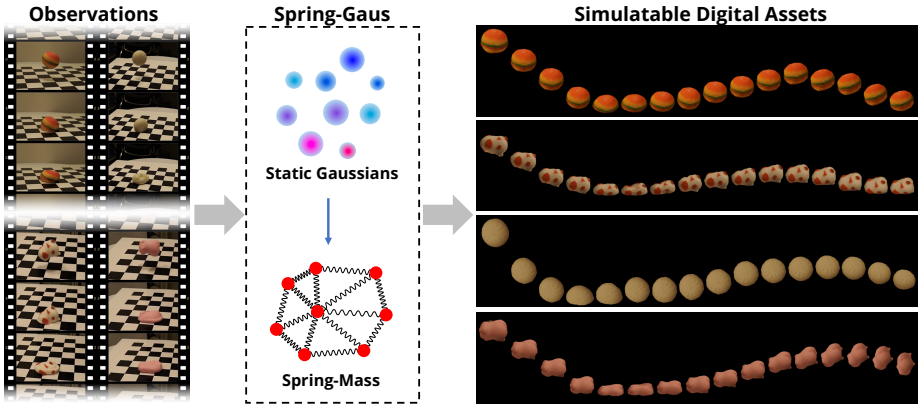

Fig. 1: Spring-Gaus reconstructs the appearance, geometry, and physical dynamic properties of elastic objects from video observations. Spring-Gaus enables future predictions and simulations under different initial states and environmental conditions.

objects; thus, they typically cannot predict the future dynamics of these objects. While a few recent approaches using MPM [11,39] have attempted to integrate physics-based priors into 3D object representations, such as PAC-NeRF [18]. Their ability to handle real, especially heterogeneous, objects is limited, as they assume a known material model and only assigns a global physical parameter to the entire object which restricts its adaptability to real objects. Assigning learnable physical parameters to each particle in an MPM is theoretically possible. However, in practice, it incurs extreme computational costs since MPM requires tens of thousands of dense points. In addition, PAC-NeRF [18] use a implicit grid representation, due to the computational demands, the resolution of the grid is limited, which can lead to a loss of detail in the appearance modeling when using real or noisy data. Thus, the core challenge in reconstructing and predicting object dynamics lies in developing and integrating an *expressive* and *efficient* physical model for the dynamics. The physical dynamics model should be expressive enough to capture the motions of elastic objects, including collisions, deformations, and bouncing. It must also be efficient and conducive to inverse parameter estimation through gradient-based optimization.

In this work, we propose Spring-Gaus, a 3D object representation that integrates a 3D Spring-Mass model. Our model represents the elastic object dynamics properties through a learnable system of mass points and springs. Our design is expressive in that it assumes a general and widely applicable physical model class. With a learnable topology and physical parameters, Spring-Gaus can model complex deformation and motion for heterogeneous elastic objects. In addition to expressiveness, Spring-Gaus is also highly efficient for the inverse optimization of physical parameters due to its differentiable nature.

Spring-Gaus enables reconstructing and simulating elastic objects from sparse multi-view videos (Fig. 1). To overcome the intrinsic difficulty in optimization,

we propose a reconstruction pipeline that decouples physical parameter reconstruction from appearance and geometry reconstruction. Spring-Gaus requires only a few multi-view videos for physical parameter identification and is robust to the quality of geometry reconstruction.

We evaluate the effectiveness of Spring-Gaus on both synthetic and real-world datasets, demonstrating its ability to accurately reconstruct and simulate elastic objects. Our Spring-Gaus allows accurate future prediction and simulations under varying initial states and environmental parameters, showcasing its potential for applications in predictive visual perception and immersive experiences. In summary, the main contributions of this work are threefold:

- We propose Spring-Gaus, which incorporates an expressive yet efficient 3D Spring-Mass model for reconstructing and simulating elastic objects.
- We introduce a pipeline to reconstruct Spring-Gaus from multi-view videos of the object. We decouple the appearance and geometry reconstruction from the physical dynamics reconstruction for more effective optimization.
- We demonstrate the effectiveness of Spring-Gaus on both synthetic and real-world datasets, showcasing accurate reconstruction and simulation of elastic objects. This includes capabilities for future prediction and simulation under varying initial configurations.

2 Related Work

3D Object Representations: Traditionally, 3D objects are often represented by point clouds, meshes, and voxels. Recently, neural 3D object representation has become popular due to the efficiency and flexibility. For example, Scene Representation Networks (SRNs) [38] and DeepSDF [29] represent significant advancements in 3D scene and object representation, treating 3D objects as continuous functions that map world coordinates to a feature representation of local object properties. Over the past few years, NeRF [28] and its successors [1, 2, 25, 47, 50] have demonstrated the efficacy of neural networks in capturing continuous 3D scenes and objects through implicit representation. DirectVoxGO [40] accelerates NeRF's approach by substituting the MLP with a voxel grid. Furthermore, 3D Gaussian Splatting [14] has emerged as a method for real-time differentiable rendering, representing scenes and objects with 3D Gaussians. Extending this approach, DreamGaussian [41] applies 3D Gaussians for 3D object generation. Unlike these approaches, our method focuses on reconstructing simulatable 3D objects. Recently, PhysGaussian [46] integrated physical simulation into 3D Gaussians using a customized Material Point Method, allowing forward simulation of reconstructed objects. In contrast to PhysGaussian, our work focuses on system identification from raw videos and supports both the forward simulation and the inverse reconstruction of physical objects.

Dynamic Scene Reconstruction: The modeling of dynamic scenes has seen significant progress with the adoption of NeRF [8, 10, 18, 20, 22, 23, 30–32, 42, 43] and 3D Gaussian [12, 16, 24, 26, 44, 48, 49, 51] representations. D-NeRF [32] introduces an extension to NeRF, capable of modeling dynamic scenes from monocu-

lar views by optimizing an underlying deformable volumetric function. Furthermore, Dynamic 3D Gaussians [26] focus on optimizing the motion of Gaussian kernels for each frame, presenting an efficient method to capture scene dynamics. Deformable 3D Gaussians [49] propose a novel approach by learning Gaussian distributions in a canonical space, complemented with a deformation field for modeling monocular dynamic scenes. Meanwhile, 4D Gaussian Splatting [44] introduces a hybrid model combining 3D Gaussians with 4D neural voxels. PAC-NeRF [18] delves into the integration of Lagrangian particle simulation with Eulerian scene representation, exploring a new aspect of scene dynamics. Among them, PAC-NeRF [18] is the most relevant to our work. PAC-NeRF learns physical parameters used in the Material Point Method (MPM) for better dynamic reconstruction and prediction. However, PAC-NeRF assumes known material models that restrict its adaptability and applicability to complex real objects. In contrast, our work integrates a 3D Spring-Mass model that is both expressive and efficient, allowing reconstructing and simulating real elastic objects.

Physics-Informed Learning: Physics-Informed Learning has emerged as a prominent research direction since the introduction of Physics-Informed Neural Networks (PINNs) [33]. Li *et al.* [21] and Sanchez-Gonzalez *et al.* [34] introduced Graph Network-based simulators within a machine learning framework. INSR-PDE [3] tackles time-dependent partial differential equations (PDEs) using implicit neural spatial representations. NCLaw [27] focuses on learning neural constitutive laws for PDE dynamics. DiffPD [9] presents a differentiable soft-body simulator. Li *et al.* [19] have made strides in learning preconditioners for conjugate gradient PDE solvers. Chu *et al.* [4] and Yu *et al.* [52] modeled smoke in neural density and velocity fields. Neural Flow Maps [7] integrate fluid simulation with neural implicit representations. Deng *et al.* [6] introduced a novel differentiable vortex particle method for fluid dynamics inference. DANO [17] is the most relevant work to ours. In particular, DANO develops a differentiable simulation for rigid objects represented by NeRFs, allowing reconstruction and simulation from a few videos. In contrast, our work focuses on elastic objects, which involve a fundamentally new set of challenges and require a redesign of the physical models.

3 Approach

We propose Spring-Gaus that integrates a 3D Spring-Mass model with 3D Gaussians for elastic object reconstruction and simulation. Specifically, we are given a set of O calibrated videos from multiple viewpoints of an elastic object in motion, denoted as $\{I^{o,f}\}_{o=1,f=1}^{O,F}$, where F is the number of video frames and $I^{o,f}$ denotes a 2D image at video o and frame f. Our goal is to reconstruct the appearance, geometry, and physical dynamics parameters of Spring-Gaus.

In the following, we first introduce the representation for the static properties, i.e., appearance and geometry, and then the dynamics properties, i.e., the 3D Spring-Mass model. Then we describe our reconstruction pipeline. We show an overview of our reconstruction pipeline in Fig. 2.

3.1 Appearance and Geometry Representation

Following 3D Gaussians Splatting [14], we use a set of 3D Gaussian kernels to represent the appearance and shape of an object. These kernels are described by a full 3D covariance matrix Σ, defined in world space and centered at point $\boldsymbol{\mu}$:

$$G(\boldsymbol{\tau}) = e^{-\frac{1}{2}(\boldsymbol{\tau})^T \Sigma^{-1}(\boldsymbol{\tau})}, \tag{1}$$

where $\boldsymbol{\mu}$ is the mean of the Gaussian distribution, and $\boldsymbol{\tau}$ is the independent variable of the Gaussian distribution. In our formulation, we assume all Gaussian kernels are isotropic, the covariance matrix Σ can be controlled by a scalar $s \in \mathbb{R}$.

The splatting process, designed to render Gaussian kernels into a 2D image, involves two main steps: (1) projecting the Gaussian kernels into camera space, and (2) rendering the projected kernels into image space. The projection process is defined as:

$$\Sigma' = JW\Sigma W^T J^T, \tag{2}$$

where Σ' represents the covariance matrix in camera coordinates, W is the transformation matrix, and J denotes the Jacobian of the affine approximation of the projective transformation. The color C of each pixel is rendered as:

$$C = \sum_{i \in N} T_i \alpha_i \boldsymbol{c}_i, \tag{3}$$

where N is the total number of kernels and T_i represents the transmittance, defined as $\Pi_{j=1}^{i-1} \alpha_j$. The term α_i denotes the alpha value for each Gaussian, which is calculated using the expression $1 - e^{-\sigma_i \delta_i}$, where σ_i is the opacity factor. Additionally, $\boldsymbol{c}_i$ refers to the color of the Gaussian along the ray within the interval δ_i.

For each kernel, the learnable parameters include a point center $\boldsymbol{\mu}$, a scaling scalar s, a color vector $\boldsymbol{c}$, and an opacity value σ.

3.2 Physics-Based Dynamics with 3D Spring-Mass Model

We aim to develop and integrate an *expressive* yet *efficient* physics-based dynamics model. With these two design goals in mind, we introduce the 3D Spring-Mass model which represents the elastic object dynamics by a learnable spring-mass system. Our Spring-Gaus does not assume any material type, and it is expressive to model real heterogeneous elastic objects. Besides being expressive, our dynamics model should also be efficient and amenable to gradient-based optimization for parameter estimation.

Recall that the appearance and geometry representations include a set of Gaussian kernels $\{G_i\}_{i=1}^N$, parameterized by point centers $\boldsymbol{X} = \{\boldsymbol{\mu}_i\}_{i=1}^N$, scaling scalars $\{s_i\}_{i=1}^N$, color vectors $\{\boldsymbol{c}_i\}_{i=1}^N$, and opacity values $\{\sigma_i\}_{i=1}^N$. To manage complexity for efficient simulation, we introduce volume sampling to generate a set of anchor points $\boldsymbol{A} = \{\boldsymbol{x}_i\}_{i=1}^{N_A}$ (each $\boldsymbol{x}$ represents a spatial point), defined by:

$$\boldsymbol{A} = \{\boldsymbol{x}_i\}_{i=1}^{N_A} = \mathcal{V}(\boldsymbol{X}), \tag{4}$$

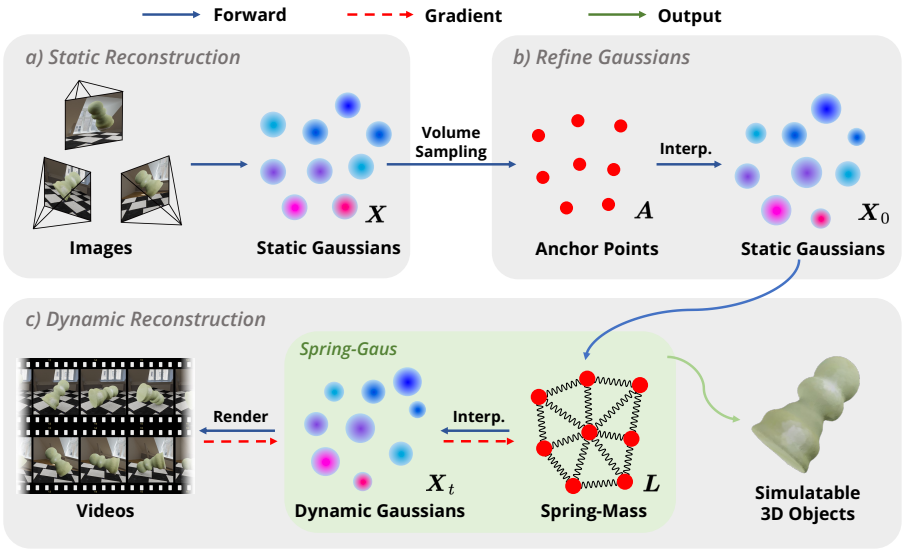

Fig. 2: Overview of Spring-Gaus reconstruction pipeline: (a) Static Scene Reconstruction: We start by reconstructing static 3D Gaussians from the first frames of the multiview videos. **(b) Refining 3D Gaussians:** We extract a set of anchor points to allow efficient simulation, which leads to appearance drift. We refine the 3D Gaussians to better model the appearance during simulation. **(c) Dynamic Reconstruction:** Our 3D Spring-Mass model simulates anchor points and updates the positions of Gaussian kernels. Upon completion of optimization, we obtain a simulatable 3D object that accurately models its dynamics.

where N_A denotes the number of anchors, and $\mathcal{V}$ denotes the sampling function.

Our 3D Spring-Mass physical model can simulate the motion of anchor points $\boldsymbol{A}$, assuming each anchor has a mass m_i and an initial velocity $\boldsymbol{v}_i$. Each anchor $\boldsymbol{x}_i$ connects to its n_k nearest neighbors $\boldsymbol{N}_i = \{\boldsymbol{x}_{i,j}\}_{j=1}^{n_k}$ through springs $\boldsymbol{L}$:

$$\boldsymbol{L} = \{l_{i,j}\}_{i=1,j=1}^{N_A,n_k} = \text{knn}(\boldsymbol{A}, \boldsymbol{A}, n_k), \tag{5}$$

where $l_{i,j}$ denotes the spring's length between $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i,j}$, and knn denotes the k-nearest neighbors function. Each spring is characterized by a stiffness $k_{i,j}$ and a damping factor $\zeta_{i,j}$.

To update the positions of the kernels, we first measure the distance between each kernel center and its n_b nearest anchors at the dynamic simulation's onset:

$$\{d_{i,j}\}_{i=1,j=1}^{N,n_b} = \text{knn}(\boldsymbol{X}, \boldsymbol{A}, n_b). \tag{6}$$

For each timestep t, the forces $\boldsymbol{F}_i^t$ acting on each anchor point $\boldsymbol{x}_i^t$ are calculated as follows:

$$\boldsymbol{F}_i^t = \boldsymbol{F}_{\boldsymbol{k}_i}^t + \boldsymbol{F}_{\boldsymbol{\zeta}_i}^t + m_i \boldsymbol{g}, \tag{7}$$

where $\boldsymbol{F_k}_i^t$, $\boldsymbol{F_\zeta}_i^t$, and $\boldsymbol{g}$ represent the spring force, the damping force, and gravitational acceleration, respectively.

Then, for each spring $\boldsymbol{L}_{i,j}$, the spring force $\boldsymbol{F_k}_{i,j}^t$ and damping force $\boldsymbol{F_\zeta}_{i,j}^t$ are determined by:

$$\boldsymbol{F_k}_{i,j}^t = -\eta_j \cdot k_{i,j} \left(\left\| \boldsymbol{x}_i^t - \boldsymbol{x}_{i,j}^t \right\| - l_{i,j} \right) \frac{\boldsymbol{x}_i^t - \boldsymbol{x}_{i,j}^t}{\left\| \boldsymbol{x}_i^t - \boldsymbol{x}_{i,j}^t \right\|} \cdot \left| \left\| \boldsymbol{x}_i^t - \boldsymbol{x}_{i,j}^t \right\| - l_{i,j} \right|^{p_k}, \quad (8)$$

$$\boldsymbol{F_\zeta}_{i,j}^t = \left(-\zeta_{i,j} \left(\boldsymbol{v}_i^t - \boldsymbol{v}_{i,j}^t \right) \frac{\boldsymbol{x}_i^t - \boldsymbol{x}_{i,j}^t}{\left\| \boldsymbol{x}_i^t - \boldsymbol{x}_{i,j}^t \right\|} \right) \cdot \frac{\boldsymbol{x}_i^t - \boldsymbol{x}_{i,j}^t}{\left\| \boldsymbol{x}_i^t - \boldsymbol{x}_{i,j}^t \right\|}, \quad (9)$$

where η is a soft vector which will be discussed later and p_k is a hyperparameter that determines the nonlinearity of the spring force. When p_k is set to 0, Eq. (8) becomes Hooke's law, and for positive values of p_k, the spring force becomes a nonlinear function of the spring's length. The cumulative forces acting on each anchor point $\boldsymbol{x}_i^t$ are expressed as:

$$\boldsymbol{F}_i^t = \sum_{j=1}^{n_k} \boldsymbol{F_k}_{i,j}^t + \sum_{j=1}^{n_k} \boldsymbol{F_\zeta}_{i,j}^t + m_i \boldsymbol{g}. \quad (10)$$

Anchor points $\boldsymbol{A}$'s positions and velocities are updated using semi-implicit Euler integration:

$$\hat{\boldsymbol{v}}_i^{t+1} = \boldsymbol{v}_i^t + \frac{\boldsymbol{F}_i^t}{m_i} \Delta t, \quad (11)$$

$$\hat{\boldsymbol{x}}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1} \Delta t, \quad (12)$$

and a boundary condition $\mathcal{B}$ is applied to the anchor points $\boldsymbol{A}$ to model the interactions with the environment:

$$\boldsymbol{x}_i^{t+1}, \boldsymbol{v}_i^{t+1} = \mathcal{B}(\hat{\boldsymbol{x}}_i^{t+1}, \hat{\boldsymbol{v}}_i^{t+1}). \quad (13)$$

The position of each Gaussian kernel $\boldsymbol{\mu}_i$ is updated through Inverse Distance Weighting (IDW) interpolation to reflect the dynamic changes accurately:

$$\boldsymbol{\mu}_i^{t+1} = \frac{\sum_{j=1}^{n_b} \boldsymbol{x}_{i,j}^{t+1} \cdot (1/(d_{i,j})^{p_b})}{\sum_{j=1}^{n_b} (1/(d_{i,j})^{p_b})}, \quad (14)$$

where p_b is a positive real number that determines the diminishing influence of anchor points with distance. To render the image $\hat{I}$ at a specific camera and frame, we use the updated positions of the Gaussian kernels following the rendering equation Eq. (3).

Soft Vector for Springs Connection: In our formulation above, the n_k is a hyperparameter which is the number of connected springs for each anchor. However, the choice of n_k will directly affect the simulation results markedly. The bigger value of n_k, the object will behave more rigidly, and a smaller value of n_k leads to a noticeably softer behavior of the point cloud. To address the

significant impact that the value of n_k has on the simulation, we introduce a mitigation strategy by applying a soft vector $\eta = [\eta_0, \eta_1, ..., \eta_{n_k}]$. This vector controlled by a learnable parameter κ (shared by all anchors) is used to modulate the number of connected springs, thereby adjusting the system's response to a different object. Given an empirical value n_c, the soft vector η is calculated as:

$$\eta_j = \begin{cases} 1 & j \leq n_c, \\ \text{clamp}(2 - \exp(\text{softplus}(\kappa))^{j-n_c}, 0, 1) & n_c < j \leq n_k. \end{cases} \tag{15}$$

3.3 Optimization

To allow efficient optimization, we simplify our model by reducing the number of learnable parameters without changing the essential expressiveness. In our simplified approach, we standardize the mass of every anchor to a constant value m_0, and control all damping factors using a singular parameter ζ_0. These two parameters are fixed, eliminating their variability from the optimization process. Furthermore, we introduce a unified parameter k_i for each anchor $\boldsymbol{x}_i$ to control the spring stiffness of the springs connected to it, simplifying the model without compromising its functional integrity. The spring stiffness $k_{i,j}$ and damping factor $\zeta_{i,j}$ are thus given as follows:

$$k_{i,j} = k_i / l_{i,j}, \tag{16}$$

$$\zeta_{i,j} = \zeta_0 / l_{i,j}. \tag{17}$$

Note that, in Eq. (17) stiffness is defined at anchor points rather than on springs themselves to simplify the optimization—we only need to optimize N_A stiffness coefficients instead of all the springs, which is on the order of $n_k \cdot N_A$. By this simplification, our model maintains computational efficiency and ease of optimization, while still capturing the essential dynamics of the object.

Summarized, the learnable parameters in our model now include:

- $\boldsymbol{v}_0$: the initial velocity vector, providing a baseline movement pattern for the simulation;
- $\{k_i\}_{i=1}^{N_A}$: the individual stiffness parameters for each anchor, allowing for localized adjustments to spring stiffness;
- κ: a parameter governing the modulation of the soft vector, facilitating fine-tuned control over the spring dynamics;
- $\Theta(\mathcal{B}(\cdot))$: parameters defining the boundary conditions (Θ is learnable parameters), such as the friction coefficient, which influence the simulation's physical realism.

There exist n_t timesteps between each of the two keyframes. At each timestep, $\boldsymbol{x}^t$ is computed from $\boldsymbol{x}^{t-1}$ using the update rule Eqs. (11) and (12). We only optimize physical parameters at each keyframe (when $t = 0, n_t, 2n_t, 3n_t, \ldots$) based on the visual observations until that time. We increase the value of n_t as the parameters converge. This approach balances identification accuracy and computational demand.

Same with 3D Gaussian Splatting [14], we define our loss function as a weighted combination of the $\mathcal{L}_1$ norm and the Structural Similarity Index Measure (D-SSIM) $\mathcal{L}_{\text{d-ssim}}$, applied between the input images I and the rendered images $\hat{I}$. This is formally expressed as:

$$\mathcal{L} = (1 - \lambda_{\text{d-ssim}})\mathcal{L}_1 + \lambda_{\text{d-ssim}}\mathcal{L}_{\text{d-ssim}}, \tag{18}$$

where $\lambda_{\text{d-ssim}}$ is a weighting coefficient that balances the contribution of the $\mathcal{L}_1$ norm and the D-SSIM term to the overall loss. We decouple our optimization into a few stages, as illustrated in Fig. 2.

Static Reconstruction: Our optimization starts by taking the first frames of the multi-view videos and using them to reconstruct the appearance and geometry of the object.

3D Gaussians Refinement: During dynamic simulation of the anchor points, the 3D Gaussian centers $\boldsymbol{X}_0 = \{\boldsymbol{\mu}_i^0\}_{i=1}^N$ are computed by IDW interpolation outlined in Eq. (14). This leads to a slight appearance drift from the static reconstruction. Therefore, we refine the parameters of the Gaussian kernels—scaling scalar s, color vector $\boldsymbol{c}$, and opacity value σ, excluding points center $\boldsymbol{\mu}$—also at the first frame, before the dynamic simulation. Since the Gaussian kernels' position from IDW interpolation is slightly different from the static reconstruction results, this refinement process enables the Gaussian kernels with the interpolated spatial position to render the correct object appearance.

Dynamic Reconstruction: Since our efficient dynamics simulation is fully differentiable with respect to the learnable parameters, we can optimize the physical parameters through differentiable simulation and differentiable rendering with our loss function $\mathcal{L}$.

4 Experiment

4.1 Datasets

Synthetic Data. We evaluate Spring-Gaus using a synthetic dataset. We first collected fourteen 3D models, including point clouds and meshes, to use as initial point clouds for our simulations. Some of them are sourced from PAC-NeRF [18] and OmniObject3D [45]. Following PAC-NeRF, we employ the Material Point Method [11,39] to simulate the dynamics of elastic objects to generate synthetic data. Our dataset features elastic objects with various stiffness levels and diverse geometric forms. Multi-view RGB videos are rendered using Blender [5], with each sequence comprising 10 viewpoints and 30 frames at a resolution of 512×512. The initial 20 frames are utilized for dynamic reconstruction, while the subsequent 10 frames are dedicated to evaluating future prediction performance.

Real-World Data. In addition to synthetic datasets, we further assess Spring-Gaus using captured real-world examples. Our collection process distinguishes between static scenes and dynamic multi-view videos. For static scenes, we position each object on a table and capture 50-70 images from various viewpoints within the upper hemisphere surrounding the object. To obtain camera

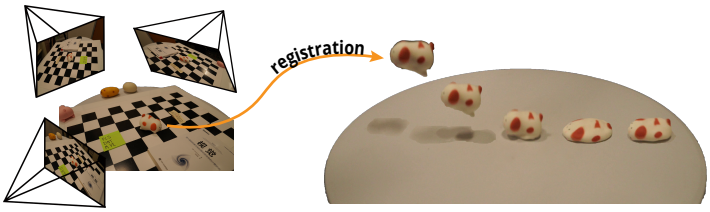

Fig. 3: Registration from static scene to dynamic scene for real-world sample.

poses for static scenes, we utilize the off-the-shelf Structure-from-Motion toolkit COLMAP [35,36]. The dynamic aspect of our dataset is represented through multi-view RGB videos, recorded from three distinct viewpoints, at a resolution of 1920×1080. The camera parameters for dynamic scenes are obtained through calibration using a checkerboard. SAM [15] is used to obtain object masks.

4.2 Implementation Details

When learning the 3D Gaussians, the distribution of Gaussian kernels is highly dependent on the initial points. Gaussian kernels tend to concentrate on the surface if the initial points are derived from SfM results. However, we initialize a large number of points inside a cube, resulting in a more uniform distribution of the final kernels.

For real-world samples, we collect static and dynamic scenes separately because three viewpoints are insufficient for effective 3D Gaussian reconstruction [14]. This results in the Gaussian kernels for static and dynamic scenes being in different coordinate systems. Therefore, for real samples, we employ a registration network before dynamic reconstruction to align the Gaussian kernels from static scene coordinates to dynamic scene coordinates, as shown in Fig. 3. Specifically, we optimize a scale factor s_r, a translation vector $\boldsymbol{t}_r$, and a rotation vector $\boldsymbol{r}_r$ for the registration. We represent 3D rotations using a continuous 6D vector $\boldsymbol{r}_r \in \mathbb{R}^6$, which has been shown to be more amenable for gradient-based optimization [53]. However, slight deformations of objects and variations in lighting conditions and exposure times during data capture are inevitable. These variations can result in color discrepancies between frames across time and viewpoints. The color discrepancy is the major noise source that hinders registration and reconstruction. Thus, we refine our approach by computing the loss function between mask images, incorporating both mask center loss and perceptual loss into our model. Consequently, the revised loss function for real samples is expressed as:

$$\mathcal{L} = (1 - \lambda_{\text{d-ssim}})\mathcal{L}_1 + \lambda_{\text{d-ssim}}\mathcal{L}_{\text{d-ssim}} + \lambda_{\text{center}}\mathcal{L}_{\text{center}} + \lambda_{\text{percep}}\mathcal{L}_{\text{percep}}, \quad (19)$$

where $\lambda_{\text{d-ssim}} = 0.8$, $\lambda_{\text{center}} = 1.0$, and $\lambda_{\text{percep}} = 0.1$ are the weighting coefficients. Here, $\mathcal{L}_{\text{center}}$ quantifies the discrepancy between the center coordinates of

		Torus	Cross	Cream	Apple	Paste	Chess	Banana	Mean
CD↓	Spring-Gaus (ours)	**2.38**	**1.57**	2.22	**1.87**	**7.03**	**2.59**	**18.48**	**5.16**
	PAC-NeRF [18]	2.47	3.87	**2.21**	4.69	37.70	8.20	66.43	17.94
EMD↓	Spring-Gaus (ours)	0.087	**0.051**	0.094	**0.076**	**0.126**	**0.095**	**0.135**	**0.095**
	PAC-NeRF [18]	**0.055**	0.111	**0.083**	0.108	0.192	0.155	0.234	0.134
PSNR↑	Spring-Gaus (ours)	16.83	**16.93**	**15.42**	**21.55**	**14.71**	**16.08**	**17.89**	**17.06**
	PAC-NeRF [18]	**17.46**	14.15	15.37	19.94	12.32	15.08	16.04	15.77
SSIM↑	Spring-Gaus (ours)	**0.919**	**0.940**	**0.862**	**0.902**	**0.872**	**0.881**	**0.904**	**0.897**
	PAC-NeRF [18]	0.913	0.906	0.858	0.878	0.819	0.848	0.866	0.870

Table 1: Quantitative results of future prediction on synthetic data. Spring-Gaus excels in short-term future prediction. Meanwhile, since we separate appearance and dynamics modeling, Spring-Gaus also maintains good rendering quality.

		Torus	Cross	Cream	Apple	Paste	Chess	Banana	Mean
CD↓	Spring-Gaus (ours)	**0.17**	**0.48**	**0.36**	**0.38**	**0.19**	1.80	**2.60**	**0.85**
	PAC-NeRF [18]	4.92	1.10	0.77	1.11	3.14	**0.96**	2.77	2.11
	Dy-Gaus [26]	579	773	479	727	2849	764	2963	1305
	4D-Gaus [44]	11.12	1.77	2.87	2.23	1.95	3.97	7.13	4.43
EMD↓	Spring-Gaus (ours)	**0.040**	**0.037**	**0.031**	**0.033**	**0.022**	0.063	**0.052**	**0.040**
	PAC-NeRF [18]	0.056	0.052	0.041	0.045	0.054	**0.052**	0.062	0.052
	Dy-Gaus [26]	0.857	0.955	0.783	0.903	1.739	0.985	1.591	1.116
	4D-Gaus [44]	0.130	0.078	0.089	0.088	0.070	0.097	0.112	0.095

Table 2: Quantitative results of dynamic reconstruction on synthetic data. Spring-Gaus has excellent geometric accuracy in dynamic reconstruction.

the rendered and ground truth images, while $\mathcal{L}_{\text{percep}}$ represents the perceptual loss [13], which is based on the VGG16 architecture [37].

4.3 Qualitative and Quantitative Results

We assess the performance of our approach using both synthetic and real-world datasets. Firstly, we validate the effectiveness of our method on synthetic datasets, utilizing the first 20 frames as our observation set for training dynamic modeling capabilities. For future prediction, we employ the subsequent 10 frames, comparing the ground truth with our model's predictions of future frames. Additionally, we benchmark our approach against the most relevant methods in dynamic scene modeling and physics-informed learning, including PAC-NeRF [18], Dynamic 3D Gaussians [26], and 4D Gaussian Splatting [44].

The qualitative results are presented in Fig. 4. We also report quantitative results by computing the Chamfer Distance (CD) and Earth Mover's Distance (EMD). In all tables, the Chamfer Distance is measured based on squared distance, with units expressed as $1 \times 10^3 mm^2$. The quantitative analysis of dynamic reconstruction, shown in Tab. 2, reveals that Spring-Gaus can accurately simulate object dynamics for the synthetic data.

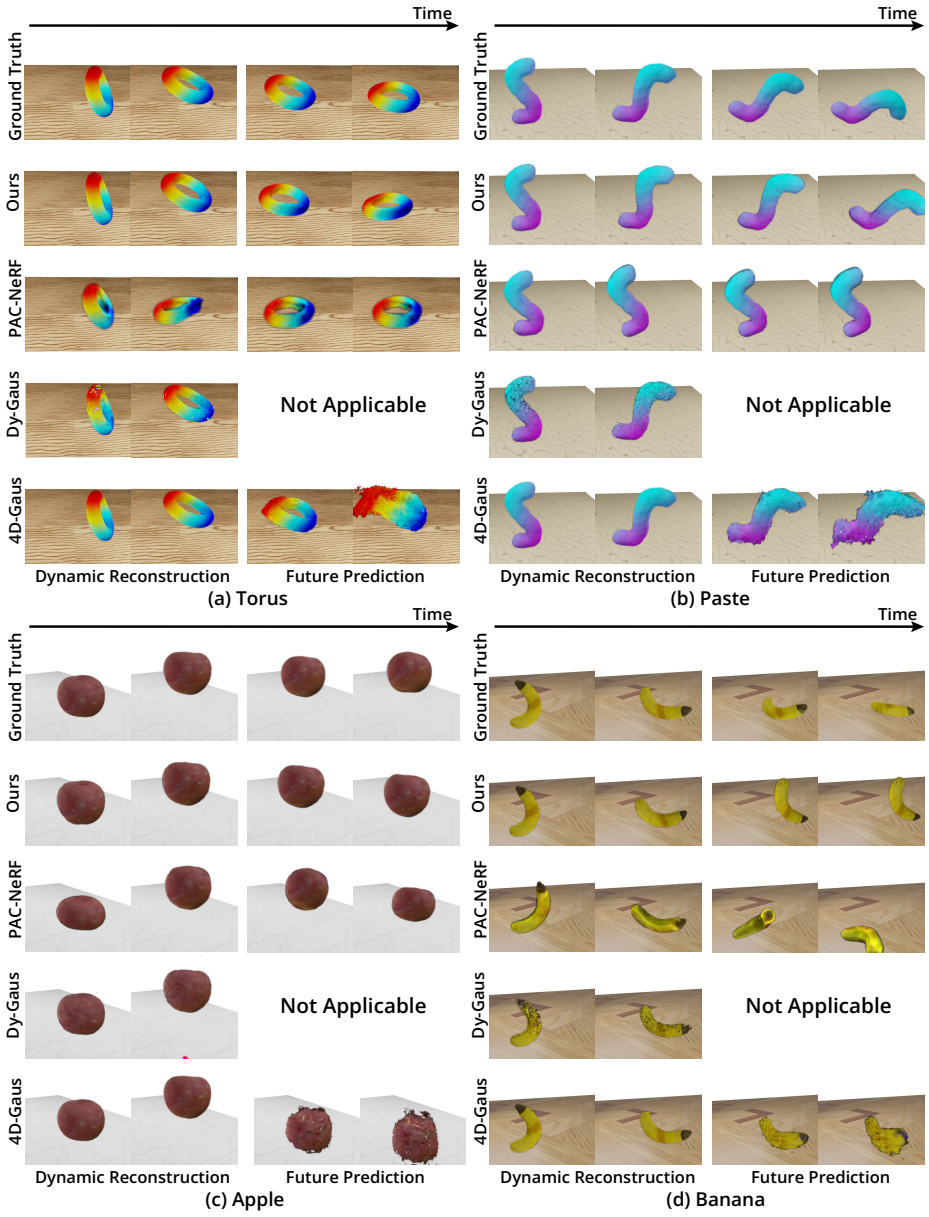

Fig. 4: Qualitative results on synthetic data. Compared with PAC-NeRF [18], Dynamic 3D Gaussians [26] and 4D Gaussian Splatting [44], Spring-Gaus can maintain a good geometry and appearance while reconstructing reasonable dynamics.

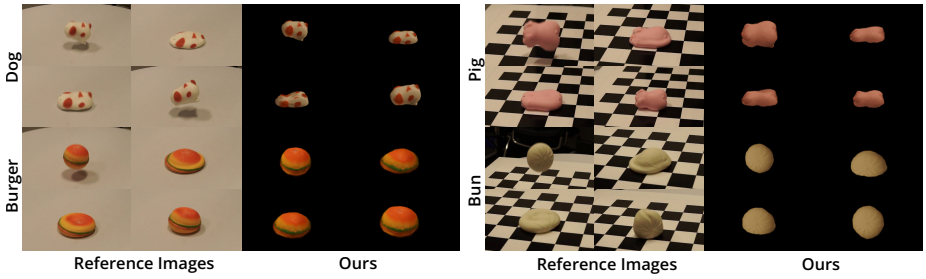

Fig. 5: Qualitative results of future prediction on real-world samples. Predicted dynamics closely follow real observations.

In Tab. 1, we show our method's capability in predicting future frames. Our method outperforms PAC-NeRF across CD, EMD, Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity Index (SSIM) metrics. We show qualitative results in Fig. 4. We observe that Spring-Gaus gives faithful reconstruction and future prediction that closely aligns with the ground truth, even though the simulation engines are different. This shows that Spring-Gaus is not only expressive but also allows efficient identification through gradient-based optimization.

We also evaluate Spring-Gaus on real-world samples. It is difficult for both NeRF [28] and 3D Gaussian Splatting [14] to reconstruct the correct geometric information under extremely sparse camera views. However, due to 3D Gaussians' explicit representation that we can directly operate Gaussian kernels, we could reconstruct the static scenes and dynamic scenes in a different coordinate system and then align them using a registration network mentioned in Sec. 4.2, which is hard to do under an implicit representation. We show the registration process in Fig. 3. Results on real-world samples can be found in Fig. 5 and Fig. 1.

4.4 Generalization to New Conditions

In addition to future prediction, we also show that our method essentially creates a digital asset of the object from the multi-view videos, allowing dynamic simulation under different unseen environmental conditions. In Fig. 6, we edit the boundary conditions, such as adjusting the positions and the stickiness of the ground plane. In Fig. 7, we edit physical conditions, initialization conditions, and environmental conditions, such as object properties (making them softer or harder), initial velocities, and environmental gravity. Please check our project website* for videos of the results for a more expressive illustration.

* https://zlicheng.com/spring_gaus

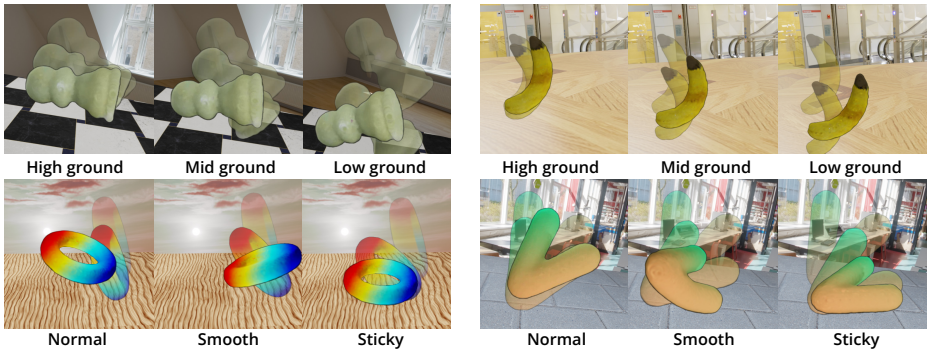

Fig. 6: We edit boundary conditions in these demos. Changing grounds' position and using smooth or sticky ground.

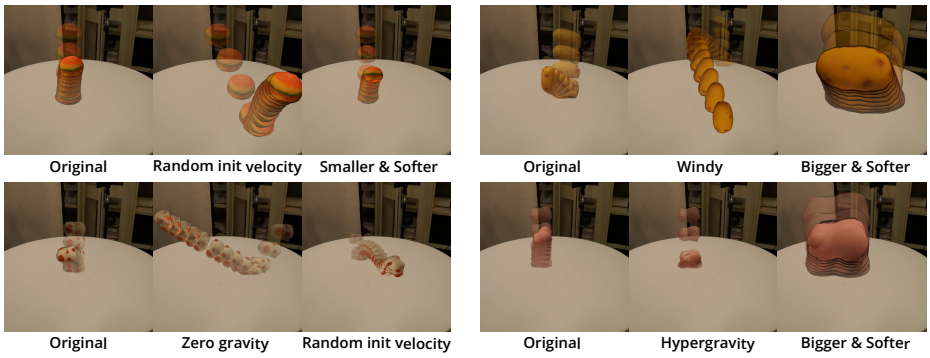

Fig. 7: We edit physical conditions, initial velocities and gravities in these demos.

5 Conclusion

In this paper, we introduce Spring-Gaus, a novel framework designed to acquire simulatable digital assets of elastic objects from video observations. By integrating a 3D Spring-Mass model, Spring-Gaus allows the reconstruction of object appearance and dynamics. A key feature of our approach is the distinct separation between the learning processes for appearance and physics, thereby circumventing potential issues with optimization interference. We evaluate Spring-Gaus on both synthetic and real-world datasets, demonstrating its capability to reconstruct geometry, appearance, and physical dynamic properties. Moreover, our method demonstrates improved capabilities by predicting short-term future dynamics under different environmental conditions. This showcases its strength in identifying physical properties from observational data and predicting the dynamics of reconstructed digital assets.

Acknowledgments. The work is in part supported by the Amazon AICE Award, NSF RI #2211258, #2338203, ONR YIP N00014-24-1-2117, and ONR MURI N00014-22-1-2740.

References

1. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. In: CVPR (2021)
2. Chen, A., Xu, Z., Zhao, F., Zhang, X., Xiang, F., Yu, J., Su, H.: Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In: ICCV (2021)
3. Chen, H., Wu, R., Grinspun, E., Zheng, C., Chen, P.Y.: Implicit neural spatial representations for time-dependent pdes. In: ICML (2023)
4. Chu, M., Liu, L., Zheng, Q., Franz, E., Seidel, H.P., Theobalt, C., Zayer, R.: Physics informed neural fields for smoke reconstruction with sparse data. ACM TOG **41**(4) (jul 2022). https://doi.org/10.1145/3528223.3530169
5. Community, B.O.: Blender - a 3D modelling and rendering package. Blender Foundation, Stichting Blender Foundation, Amsterdam (2018), http://www.blender.org
6. Deng, Y., Yu, H.X., Wu, J., Zhu, B.: Learning vortex dynamics for fluid inference and prediction. In: ICLR (2023)
7. Deng, Y., Yu, H.X., Zhang, D., Wu, J., Zhu, B.: Fluid simulation on neural flow maps. ACM TOG **42**(6) (2023)
8. Driess, D., Huang, Z., Li, Y., Tedrake, R., Toussaint, M.: Learning multi-object dynamics with compositional neural radiance fields. In: Conference on Robot Learning. pp. 1755–1768. PMLR (2023)
9. Du, T., Wu, K., Ma, P., Wah, S., Spielberg, A., Rus, D., Matusik, W.: Diffpd: Differentiable projective dynamics. ACM TOG **41**(2) (nov 2021). https://doi.org/10.1145/3490168
10. Guan, S., Deng, H., Wang, Y., Yang, X.: Neurofluid: Fluid dynamics grounding with particle-driven neural radiance fields. In: ICML (2022)
11. Hu, Y., Fang, Y., Ge, Z., Qu, Z., Zhu, Y., Pradhana, A., Jiang, C.: A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. ACM TOG **37**(4) (jul 2018). https://doi.org/10.1145/3197517.3201293
12. Huang, Y.H., Sun, Y.T., Yang, Z., Lyu, X., Cao, Y.P., Qi, X.: Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. CVPR (2024)
13. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: ECCV (2016)
14. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM TOG **42**(4) (July 2023)
15. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., Dollár, P., Girshick, R.: Segment Anything. arXiv preprint arXiv:2304.02643 (2023)
16. Kratimenos, A., Lei, J., Daniilidis, K.: Dynmf: Neural motion factorization for real-time dynamic view synthesis with 3d gaussian splatting. arXiv preprint arXiv:2312.00112 (2023)
17. Le Cleac'h, S., Yu, H.X., Guo, M., Howell, T., Gao, R., Wu, J., Manchester, Z., Schwager, M.: Differentiable physics simulation of dynamics-augmented neural objects. IEEE Robotics and Automation Letters **8**(5), 2780–2787 (2023)

18. Li, X., Qiao, Y.L., Chen, P.Y., Jatavallabhula, K.M., Lin, M., Jiang, C., Gan, C.: PAC-NeRF: Physics augmented continuum neural radiance fields for geometry-agnostic system identification. In: ICLR (2023)
19. Li, Y., Chen, P.Y., Du, T., Matusik, W.: Learning preconditioners for conjugate gradient pde solvers. In: ICML (2023)
20. Li, Y., Li, S., Sitzmann, V., Agrawal, P., Torralba, A.: 3d neural scene representations for visuomotor control. In: Conference on Robot Learning. pp. 112–123. PMLR (2022)
21. Li, Y., Wu, J., Tedrake, R., Tenenbaum, J.B., Torralba, A.: Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. arXiv preprint arXiv:1810.01566 (2018)
22. Li, Z., Niklaus, S., Snavely, N., Wang, O.: Neural scene flow fields for space-time view synthesis of dynamic scenes. In: CVPR (2021)
23. Li, Z., Wang, Q., Cole, F., Tucker, R., Snavely, N.: Dynibar: Neural dynamic image-based rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4273–4284 (2023)
24. Lin, Y., Dai, Z., Zhu, S., Yao, Y.: Gaussian-flow: 4d reconstruction with dynamic 3d gaussian particle. arXiv preprint arXiv:2312.03431 (2023)
25. Liu, Y., Peng, S., Liu, L., Wang, Q., Wang, P., Christian, T., Zhou, X., Wang, W.: Neural rays for occlusion-aware image-based rendering. In: CVPR (2022)
26. Luiten, J., Kopanas, G., Leibe, B., Ramanan, D.: Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In: 3DV (2024)
27. Ma, P., Chen, P.Y., Deng, B., Tenenbaum, J.B., Du, T., Gan, C., Matusik, W.: Learning neural constitutive laws from motion observations for generalizable pde dynamics. In: ICML (2023)
28. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NeRF: Representing scenes as neural radiance fields for view synthesis. In: ECCV (2020)
29. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: CVPR (2019)
30. Park, K., Sinha, U., Barron, J.T., Bouaziz, S., Goldman, D.B., Seitz, S.M., Martin-Brualla, R.: Nerfies: Deformable neural radiance fields. In: ICCV (2021)
31. Park, K., Sinha, U., Hedman, P., Barron, J.T., Bouaziz, S., Goldman, D.B., Martin-Brualla, R., Seitz, S.M.: Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. arXiv preprint arXiv:2106.13228 (2021)
32. Pumarola, A., Corona, E., Pons-Moll, G., Moreno-Noguer, F.: D-NeRF: Neural Radiance Fields for Dynamic Scenes. In: CVPR (2021)
33. Raissi, M., Perdikaris, P., Karniadakis, G.: Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics **378**, 686–707 (2019). https://doi.org/https://doi.org/10.1016/j.jcp.2018.10.045
34. Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., Battaglia, P.W.: Learning to simulate complex physics with graph networks. In: ICML (2020)
35. Schönberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: CVPR (2016)
36. Schönberger, J.L., Zheng, E., Pollefeys, M., Frahm, J.M.: Pixelwise view selection for unstructured multi-view stereo. In: ECCV (2016)
37. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
38. Sitzmann, V., Zollhöfer, M., Wetzstein, G.: Scene representation networks: Continuous 3d-structure-aware neural scene representations. In: NeurIPS (2019)

39. Stomakhin, A., Schroeder, C., Chai, L., Teran, J., Selle, A.: A material point method for snow simulation. ACM TOG **32**(4) (jul 2013). https://doi.org/10.1145/2461912.2461948
40. Sun, C., Sun, M., Chen, H.: Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In: CVPR (2022)
41. Tang, J., Ren, J., Zhou, H., Liu, Z., Zeng, G.: Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. ICLR (2024)
42. Tretschk, E., Tewari, A., Golyanik, V., Zollhöfer, M., Lassner, C., Theobalt, C.: Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In: ICCV (2021)
43. Wang, C., MacDonald, L.E., Jeni, L.A., Lucey, S.: Flow supervision for deformable nerf. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 21128–21137 (2023)
44. Wu, G., Yi, T., Fang, J., Xie, L., Zhang, X., Wei, W., Liu, W., Tian, Q., Xinggang, W.: 4d gaussian splatting for real-time dynamic scene rendering. In: CVPR (2024)
45. Wu, T., Zhang, J., Fu, X., Wang, Y., Ren, J., Pan, L., Wu, W., Yang, L., Wang, J., Qian, C., Lin, D., Liu, Z.: OmniObject3D: Large-vocabulary 3d object dataset for realistic perception, reconstruction and generation. In: CVPR (2023)
46. Xie, T., Zong, Z., Qiu, Y., Li, X., Feng, Y., Yang, Y., Jiang, C.: Physgaussian: Physics-integrated 3d gaussians for generative dynamics. In: CVPR (2024)
47. Xu, Q., Xu, Z., Philip, J., Bi, S., Shu, Z., Sunkavalli, K., Neumann, U.: Point-NeRF: Point-based neural radiance fields. In: CVPR (2022)
48. Yang, Z., Yang, H., Pan, Z., Zhang, L.: Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. In: ICLR (2024)
49. Yang, Z., Gao, X., Zhou, W., Jiao, S., Zhang, Y., Jin, X.: Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In: CVPR (2024)
50. Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelNeRF: Neural radiance fields from one or few images. In: CVPR (2021)
51. Yu, H., Julin, J., Milacski, Z.A., Niinuma, K., Jeni, L.A.: Cogs: Controllable gaussian splatting. arXiv preprint arXiv:2312.05664 (2023)
52. Yu, H.X., Zheng, Y., Gao, Y., Deng, Y., Zhu, B., Wu, J.: Inferring hybrid neural fluid fields from videos. Advances in Neural Information Processing Systems (2023)
53. Zhou, Y., Barnes, C., Lu, J., Yang, J., Li, H.: On the continuity of rotation representations in neural networks. In: CVPR (2019)

Appendix

A Data flow

To clarify the decoupling of appearance learning and physical learning, we provide a simpler layout of our pipeline, as shown in Fig. 8.

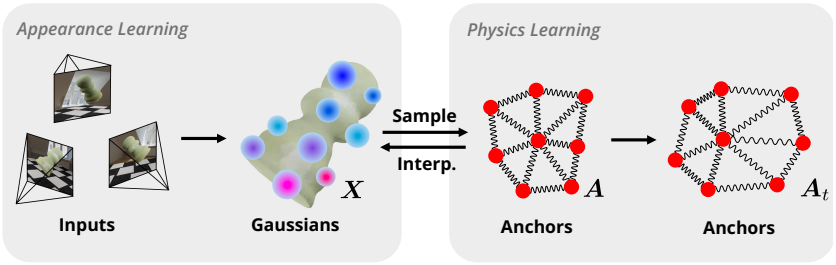

Fig. 8: Simpler layout of the pipeline. In appearance learning, we optimize Gaussian kernels for static image rendering. We sample anchors $\boldsymbol{A}$ from Gaussians' center $\boldsymbol{X}$. In physics learning, we optimize physical parameters among anchors $\boldsymbol{A}$. After simulation, at time t, the updated Gaussians' center $\boldsymbol{X}_t$ could be interpolated from $\boldsymbol{A}_t$.

B More Implementation Details

We use a single NVIDIA RTX 3090 GPU for reconstruction. For static scene reconstruction, we follow the configuration prescribed by 3D Gaussian Splatting [14], which takes approximately 10 minutes to optimize for the synthetic data. The dynamic reconstruction takes 300 iterations. We employ 2048 anchor points for our Spring-Mass model, with each anchor linked to $n_k = 256$ neighbors through springs. We set $n_b = 16$ and $n_c = 16$. For each sequence, we assume mass $m_0 = 1$ and damping factor $\zeta_0 = 0.1$. The weighting coefficient for the D-SSIM term λ_{d-ssim} is set to 0.2 for static reconstruction and 0.05 for dynamic reconstruction. In our experiments, we use a nonlinear spring force, setting $p_k = 0.5$, and for Inverse Distance Weighting (IDW) interpolation, we arbitrarily choose $p_b = 0.5$.

Following the practice from PAC-NeRF [18], we first independently optimize the initial velocity vector $\boldsymbol{v}_0$, utilizing only a few frames captured before the object interacts with the environment.

In terms of the Gaussian kernels' parameters, we optimize all of them during static scene reconstruction while maintaining a constant scaling scalar s_0 for all kernels. We have found that uniform scaling across all kernels in static scene reconstruction results in a more evenly distributed point cloud and anchor points. This consistency markedly improves the dynamic model's simulation capabilities by making the kernels' spatial distribution more uniform. It is important to note

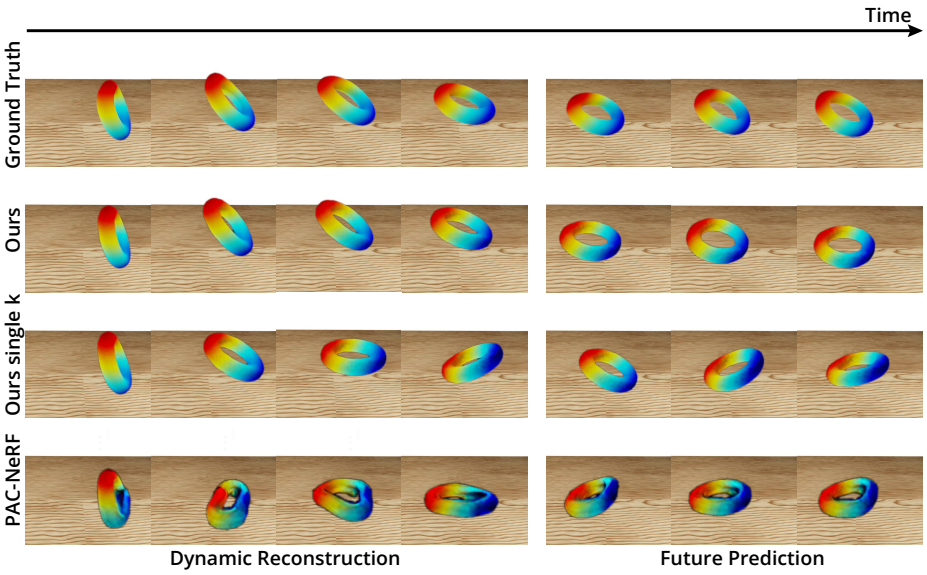

Fig. 9: Ablation study of the effectiveness of optimizing physical parameters for each particle rather than optimizing a single global parameter, on a heterogeneity object. The results shows that optimizing a single global parameter is not able to accurately model objects with complex physical properties.

that we do *not* preserve the scaling scalar as constant s_0 during this refinement phase. Instead, we assign a unique scaling scalar s_i to each Gaussian kernel associated with each anchor point.

C Ablation Study

	Dynamic Reconstruction			Future Prediction		
	CD↓	PSNR↑	SSIM↑	CD↓	PSNR↑	SSIM↑
Spring-Gaus (ours)	**0.18**	**27.08**	**0.967**	**2.04**	**17.63**	**0.927**
Spring-Gaus w/o soft vector η	0.56	25.36	0.959	13.28	13.91	0.881
Spring-Gaus, single k	3.22	23.02	0.940	6.56	14.45	0.892
PAC-NeRF [18]	8.66	19.87	0.916	5.70	15.65	0.894

Table 3: Ablation study. We demonstrate the importance of optimizing parameters for each anchor point individually as well as using a soft vector η. Optimizing parameters for each anchor point allows Spring-Gaus to have a higher degree of freedom in modeling physics, and the soft vector η gives a more flexible formulation.

In our approach, we employ a soft vector η to dynamically regulate both the quantity and intensity of springs linked to the anchor points. This strategy is illustrated in Tab. 3, showcasing its effectiveness. Our method's capability to simulate using very sparse anchors allows for the individual optimization of physical parameters for each anchor point. This contrasts with PAC-NeRF, which utilizes tens of thousands of particles, making it challenging to optimize the physical parameters for each particle infeasible. Consequently, PAC-NeRF faces limitations in accurately modeling objects composed of heterogeneous materials. In contrast, our methodology is adept at handling such complexities. As depicted in Fig. 9 and Tab. 3, we present the outcomes on a heterogeneous object that is segmented into various sections, each with distinct physical properties, thereby demonstrating our model's superior adaptability in capturing the nuanced dynamics of objects with variable material composition.

D Limitations and Future Work

Currently, Spring-Gaus is constrained to modeling elastic objects due to fixed spring lengths in our formulation; these lengths are constants established at the onset of dynamic simulation. Future work should aim to incorporate plastic deformation into the framework. This would involve developing a method to dynamically adjust the original lengths of the springs, also can make and break spring relationships during the simulation, allowing for the accurate modeling of materials that exhibit both elastic and plastic behavior.

Besides, our method focuses on simulating a single object colliding with the ground surface, while multi-object interaction is a fascinating topic but requires new model design (e.g., establishing new springs) and a more thorough evaluation under various challenging scenarios, which we identify as an excellent direction to expand our method. Other directions include considering more complicated boundary conditions and external actions.

Lastly, for better evaluation, a more comprehensive real world dataset with high spatial and temporal resolutions should be collected, including more diverse objects, materials, and interactions. This dataset should also include more challenging scenarios, such as occlusions, lighting changes, and camera motion. This kind of dataset will help to evaluate the robustness and generalization of related methods.