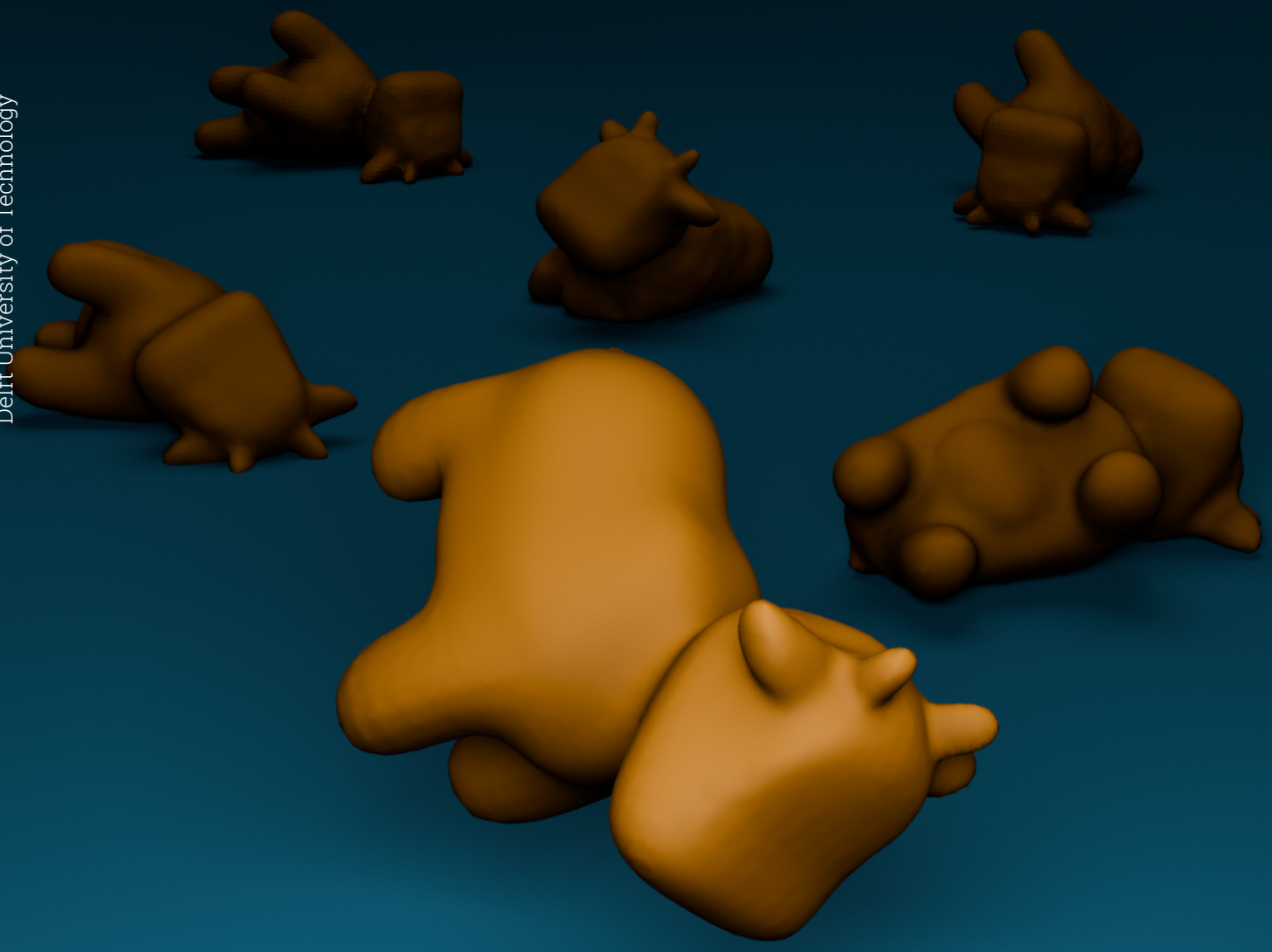
Gaussians as Supervision for Joint Physical Parameter Estimation and Appearance Reconstruction of Elastic Objects

MSc. Computer Science

Pavlos Makridis



Delft University of Technology

TU Delft

GAUSSIANS AS SUPERVISION FOR JOINT PHYSICAL PARAMETER ESTIMATION AND APPEARANCE RECONSTRUCTION OF ELASTIC OBJECTS

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

by
Pavlos Makridis
June 2024

Department of Computer Science
Faculty of EEMCS
Delft University of Technology

Pavlos Makridis: *Gaussians as Supervision for Joint Physical Parameter Estimation and Appearance Reconstruction of Elastic Objects,* © June 2024
SUPERVISORS:
Lukas Uzolas
Dr. Petr Kellnhofer
Prof. dr. Elmar Eisemann

ABSTRACT

Recovering the appearance and physical parameters of elastic objects from multi-view video is essential for many applications that require simulation of the real world. Past methods for this task have provided accurate results in recovering physical properties; however, their reliance on Neural Radiance Fields (NeRFs) for novel view synthesis means they trade off visual quality for rendering speed. To address this issue, we present a novel framework for the joint appearance reconstruction and physical parameter estimation of elastic objects relying on 3D Gaussian splatting. Our key insight is that dynamic 3D Gaussian kernels extracted from multi-view video can be used to reconstruct the object's geometry and supervise elastic parameter fitting through a differentiable physics engine. Novel views and object behaviours can then be constructed by forward simulating the extracted mesh and using it to drive the Gaussian kernels. We demonstrate that our method is competitive with the state of the art on physical parameter estimation while being better at reconstructing object appearance. Additionally, our method can simulate novel views and object interactions at near real-time rates that outperform past approaches.

ACKNOWLEDGEMENTS

First, I would like to express my sincerest gratitude to my supervisors, Lukas Uzolas and Dr. Petr Kellnhofer, for guiding me throughout this work and always being available to help. In addition, I would like to thank Professor Elmar Eisemann for his advice on this work and his teachings throughout my journey in Computer Graphics. Further, I must thank my friend Rodrigo Alvarez Lucendo for providing feedback and keeping me company in many late-night work sessions and my friend Jorge Romeu Huidobro for the same reasons and for always being up to discuss radiance fields. Last but certainly not least, I am forever grateful to my parents for their encouragement and support throughout my university years and to my friends back home for reminding me that, from time to time, it is good to live real life and not just simulate it.

So long, and thanks for all the kernels,
Pavlos Makridis
Delft, June 2024

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

FEM Finite Element Method

MPM Material Point Method

MLP Multi-Layer Perceptron

MVS Multi-View Stereo

NeRF Neural Radiance Fields

NVS Novel View Synthesis

SfM Structure-from-Motion

3DGS 3D Gaussian Splatting

1 INTRODUCTION

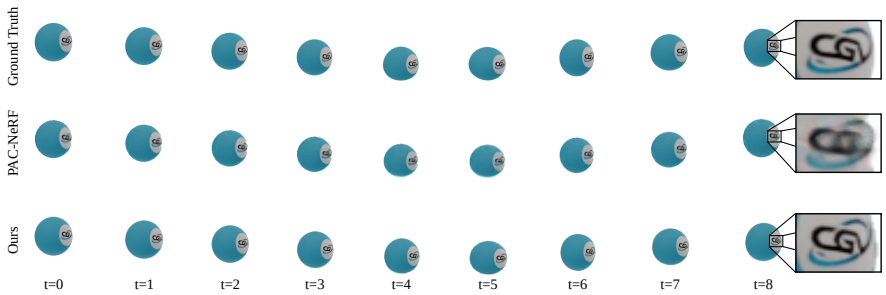


Figure 1: We present a novel framework for joint appearance reconstruction and physical parameter estimation of elastic objects. Our method works on multi-view video and uses dynamic 3D Gaussian splatting and differentiable Neo-Hookean simulation to reconstruct and simulate the recorded object. Overall, the resulting technique can recover physical parameters at the level of state-of-the-art methods while providing higher visual quality and inference speed.

Elastic objects are common in everyday life and must, therefore, be included in any applications that require real-world simulations, such as movie and video game production, virtual reality and robotic control. For these tasks, it is often needed to render the object from any viewpoint and simulate it in various scenarios. Thus, we require techniques that can reconstruct elastic objects' appearance and behaviour (dynamics). Nonetheless, achieving these goals is challenging due to the complex dynamics elastic objects demonstrate.

Traditional approaches that have tackled the above problem have relied on point cloud scanning and tracking [WWY*15], often in combination with robotic actuators that are required to accurately deform the object being scanned [CZB23; SLK*20]. These techniques have high accuracy but are largely inaccessible due to the required equipment. Simultaneously, Neural Radiance Fields (NeRF) [MST*20] and 3D Gaussian splatting (3DGS) [KKLD23] have demonstrated that high-quality appearance and geometry reconstruction from multi-view images can be achieved. However, most methods based on these new techniques are unsuitable for our purpose since they focus on capturing static scenes [BMT*21; BMV*22; KJJ*21] or reconstructing observed dynamics [XHKK21; PCPMMN21; GSKH21] and cannot make inferences about the reconstructed object's behaviour under new conditions.

To be able to generalize to novel object behaviours, a few techniques have attempted to jointly recover objects' appearance and physical elasticity parameters directly from multi-view video. The idea behind those methods is to recover object representations that can be rendered and simulated accurately using the recovered parameters. The first to try this was Virtual Elastic Objects (VEO) [CTS*22], which was later followed by PAC-NeRF [LQC*23]. Both of these techniques are NeRF-based, and they entangle the physics and appearance representation. As a result, they are prone to rendering artifacts due to errors in the physical parameter estimation and, most importantly, trade visual quality for rendering speed. More recently, Spring-Gaus [ZYWL24] attempted to mitigate these issues by utilizing the explicit radiance field representation used by 3DGS. Their method models objects as spring-mass networks of spatially varying spring stiffness and drives the Gaussians based on how the masses move. Although this approach is performant and can achieve excellent visual quality, its reliance on a spring-mass network makes it unrealistic, thus making the method unsuitable for physically accurate material characterization.

In this work, we address the limitations of the current techniques for jointly recovering the appearance and physical parameter estimation of elastic objects by combining 3DGS with differentiable Neo-Hookean elasticity simulation. Our key insight is that optimizing the Gaussians across time can create a dynamic point cloud representation of the observed deformation. This point cloud can be used directly as 3D supervision in a differentiable physics pipeline and for object reconstruction. As a result, our pipeline can produce novel views and simulate novel interactions with observed objects at near real-time rates.

1.1 SUMMARY OF CONTRIBUTIONS

We present a method combining 3D Gaussian splatting and inverse physics to achieve novel view and object interaction synthesis. We find that our method can generally estimate physical parameters at the same level of precision as state-of-the-art approaches, while being better at reconstructing object appearance. Moreover, our method can produce novel simulations and renders of the observed object faster than previous approaches, achieving near real-time rates. In summary we make the following contributions:

1. We propose a pipeline that combines 3D Gaussian splatting with Neo-Hookean differentiable simulation for appearance reconstruction and physical parameter estimation.

2. We achieve competitive results with other state-of-the-art techniques on physical parameter estimation.

3. We accomplish better appearance reconstruction results than other methods, maintaining details such as high-frequency textures even under strong object deformations.

4. We reach near real-time inference rates, outperforming past methods.

1.2 OUTLINE

This report is structured in six chapters. We start by giving an overview of related works and the theoretical background required in Chapter 2 and Chapter 3, respectively. Afterwards, we present our proposed method in Chapter 4. We then evaluate our method in Chapter 5 and present our conclusions in Chapter 6.

2

RELATED WORKS

This chapter provides an overview of the relevant literature for this work. Namely, we will be covering the three focus areas of our work: (dynamic and controllable) novel view synthesis, physical characteristics estimation and their combination. Note that this chapter aims to survey the different subfields and highlight the relevant literature. Later, in Chapter 3, we dive deeper into the theory behind the techniques presented here.

2.1 NOVEL VIEW SYNTHESIS

Novel View Synthesis (NVS) refers to synthesising images from arbitrary viewpoints of an object or scene given a set of observed images of the same object or scene [ZTS*16] (Figure 2). Early work on NVS was based on dense light fields [GGSC23], a requirement which was later relaxed with the introduction of methods based on Structure-from-Motion (SfM) [SF16] and multi-view stereo [SCD*06] which were able to reconstruct scenes from sparse sets of images. These methods, however, require blending and reprojection of the input images, which leads to high memory consumption since all input images must be stored in GPU memory and can produce blending artifacts [TFT*20; KKLD23]

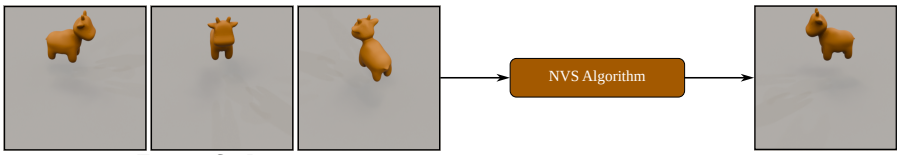


Figure 2: High level overview of Novel View Synthesis (NVS). Given a training set of images, NVS algorithms aim to render previously unseen views of the scene.

More recently, NVS has benefitted from the introduction of deep learning-based methods. Initially, deep learning was used to enhance the existing approaches, for instance, by estimating blending weights [HPP*18] or directly predicting pixel values [FNPS16]. However, these approaches still require high-quality scene modelling whose automated acquisition remains an open research problem [TFT*20]. To address this, neural rendering methods have arisen as a promising alternative.

Neural rendering methods extract an implicit representation of the underlying scene from the input data, which can be rendered. Specif-

ically, the advent of Neural Radiance Fields (NeRF) [MST*20] demonstrated that Multi-Layer Perceptrons (MLPs) combined with volumetric ray-marching could effectively be used for NVS. Many publications followed after NeRF trying to improve the visual quality of the outputs [BMT*21; BMV*22] and its rendering speed [WSND*23; MESK22; KJJ*21]. Although impressive steps have been taken to improve NeRFs, the inherent reliance on large MLPs and volumetric ray marching causes such techniques to have a substantial tradeoff between visual quality and rendering times, making them unsuitable for interactive applications.

To create a method that can achieve state-of-the-art visual quality at interactive frame rates, Kerbl and colleagues [KKLD23] proposed to use 3D Gaussian Splatting (3DGS). Their method relies on differentiable rasterisation in combination with traditional Gaussian kernel splatting [ZPVBG02]. In this way, no expensive ray-marching or large MPL is required, making this method train and render significantly faster than NeRF-based approaches. Additionally, the explicit representation used by 3DGS lends itself nicely to various downstream tasks, like colour editing or object moving, which are otherwise difficult in methods based on implicit representations.

Following the success of 3DGS, many papers have aimed to improve it by reducing the amount of memory required [PKK*24], removing rendering artifacts [RSP*24; FCC*24] or improving the ability to extract high-quality meshes from the Gaussians [GL23]. In our work, we benefit from using 3DGS since it provides high-quality appearance reconstruction and an explicit representation that can be combined with physical simulation.

2.1.1 *Dynamic Novel View Synthesis*

Dynamic NVS adds a temporal dimension to the traditional NVS problem so that dynamic scenes can be reconstructed and rendered across time. NeRFs can be adjusted for dynamic capture and generation by directly adding a temporal dimension and treating radiance over space and time [XHKK21; LNSW21]. Alternatively, other methods learn deformations of a canonical static NeRF [PCPMMN21; PSB*21; FYW*22; PSH*21] or the flow of the representation from one frame to the next [GSKH21; WELG21].

Many methods have emerged with the recent introduction of 3DGS, aiming to use this representation for dynamic NVS. Here, some approaches rely on deep learning methods to learn the observed deformation model and then use that to drive the Gaussians [DMY*23; DWY*23; WYF*23; KLD23]. Contrary to those, other methods rely only on optimizing Gaussians by either reformulating them in four dimensions [YYPZ23] or optimizing their positions and rotations across time [LKLR24]. The latter method - Dynamic 3D Gaussians - is most

relevant to our work. Using physically inspired regularisation functions, they propose an elegant approach for optimizing the positions and rotations of 3D Gaussian kernels created from the first frame. As a result, they can densely track objects and reconstruct their dynamic appearance. We are building on this method by utilizing it in the first stage of our pipeline, as it is simple and very effective in our setting.

The methods above can give great results in motion reconstruction and novel viewpoint rendering. However, they have limited to no understanding of the underlying scene dynamics and thus cannot be used to extend the observed motions. To generate novel dynamics techniques like Pie-NeRF [FSL*23] and PhysGaussian [XZQ*23], directly integrate physics-based simulations with NeRF and Gaussian splatting, respectively. In contrast, other techniques aim to use proxies from which the deformation can be transferred to the radiance representation. Such proxies can be parametric models of humans [LMR*23; PCG*19], articulated skeletons [UEK24; YHL*22] and (cage) meshes [YSL*22; PYL*22; JKK*23; BKY*22b]. The same (cage) mesh-based deformation can be easily adapted to work on 3D Gaussians as demonstrated in [GYZ*24; WBT*24; JYX*24; ZBS*23]. Inspired by classical work on simulation, which simulates deformation on tetrahedral meshes and then uses them as cages to deform a triangular mesh [MMC16], we simulate the deformation of a tetrahedral cage and similar to the techniques mentioned above, we then transfer the deformation to the Gaussians.

2.2 PHYSICAL CHARACTERISTICS ESTIMATION

System identification aims to learn mathematical descriptions of dynamic systems from input data to predict new data points from previous observations [PAG*23]. A plethora of different approaches have been developed from this, ranging from classical (statistical) kernel fitting-based methods [SL19] to those that rely on physics-informed neural networks [WJX*22]. Inverse simulation is closely related to system identification. Here, the goal is to recover the initial conditions of a system, given the evolution of this system across time [MS00]. We recognise two ways in which methods that tackle the above can be classified: based on their representation of the system and based on the supervision they require.

2.2.1 *System Representation*

We separate approaches for physical characteristics estimation depending on whether they use an *implicit* or an *explicit* system representation[1].

Implicit approaches to inverse physics aim to learn an embedding of the system's states that can be used to infer system parameters or future system states. A line of work approaches this by by learning Hamiltonian [GDY19; TRJ*19], Lagrangian [CGH*20] or more general non-linear operators [LJP*21] that can explain the dynamic behaviours of the observed system. Specifically for the case of deformable objects some approaches propose using graph and dynamic interaction neural networks for learning the system's dynamics [SGGP*20; PFSGB20; LWT*18], while other methods [XWZ*19] encodes objects in a dense latent space that allows for direct extraction of physical properties. Such methods require few prior assumptions about the physical system they model and thus they generalize to a wide variety of different systems. However, since they rely on complex neural networks they can lack interpretability, have limited predictive horizons and require large amounts of data for training. Explicit approaches address these limitations.

Explicit approaches to inverse physics tackle the problem by assuming a parameterisable physical model that describes the observed system and fitting the parameters that minimise the error between the predictions and the observations. This minimisation can be done with gradient-free iterative methods [WWY*15; SLK*20], or as is now more prevalent, by utilising differentiable physics engines. Starting with an initial (potentially random) estimate of the relevant parameters, the physics engine evolves the system through time. The resulting states are then compared with the observed ones, and the error is backpropagated through the engine to adjust the initial parameters (see Figure 3 for a high-level overview of this process). Through the recent advancements in autodifferentiation frameworks, differentiable engines have been made for a diverse set of dynamic models, like extended position-based and projective dynamics [MMC16; SC23; BML*23; DWM*21] or the Material Point Method (MPM) [HFG*18; HLS*19]. Based on these, further methods have been developed that incorporate physics engines into Deep-Learning pipelines to act as strong inductive priors [JBH19] or to be combined with inverse rendering [MMG*20]. Such explicit approaches reduce the space of learnable parameters and provide interpretable results.

In our work, we will be using the differentiable physics engine DiffSim [MMG*20], which provides a fully explicit FEM-based model

1 In the literature of physical simulation, one often encounters the terms implicit and explicit when referring to time-integration schemes. That is not their use here as we focus solely on how the system is modelled.

for simulating deformable solids based on the Neo-Hookean elasticity laws.

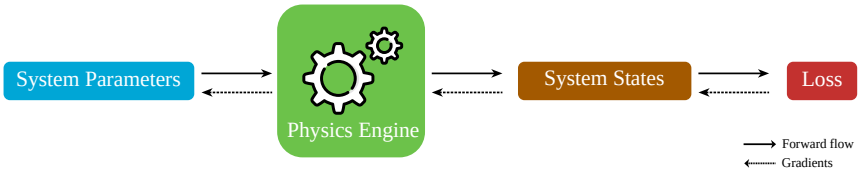


Figure 3: The general differentiable physics pipeline. System parameters are used to evolve the system and the resulting states are compared with ground truth observations through a loss function. Then, since the process is differentiable, error gradients can propagate back and adjust the system parameters.

2.2.2 *Supervision*

Another criterion for separating physical characteristics estimation techniques is the type of supervision they require. The form of supervision can vary greatly depending on the modelled system. For this work, we focus on the case of deformable solids, where there are two main approaches to supervision.

In the most straightforward case, 3D state supervision can be provided, for example, as an animated 3D mesh or point cloud. This approach was taken by many of the earlier methods in the field of physical parameter estimation [DHD*19; SC23; DWM*21; HLS*19], since it can provide an easy-to-use, strong and stable learning signal which eliminates the ambiguities that arise during image formation, however, as [MMG*20] points out, getting those 3D labels has traditionally been labour-intensive and sometimes even infeasible.

To overcome the issues associated with acquiring 3D labels, more recently, the focus has been on using 2D supervision by combining inverse physics with either neural image synthesis [JBH19] or inverse rendering [MMG*20]. Such methods have the advantage of working on video, which is often easy to acquire. This property makes such methods more practical, however, their performance can suffer compared to methods that use full 3D supervision [MMG*20].

Our approach combines the merits of both types by utilising the 3D Gaussians to automatically extract 3D states from images and then use those to optimise the physical parameters. We explain these further in Chapter 4.

2.3 JOINT APPEARANCE AND PHYSICS RECONSTRUCTION

Few approaches attempt to unify the appearance and physical characteristics estimation problem. To the best of our knowledge, the first to attempt this was [CTS*22], which combined the non-rigid

NeRF from [TTG*21] with a differentiable mesh-free forward simulator for Neo-Hookean dynamics, for which they sampled particles directly from the reconstructed NeRF volume. Similar to this, PAC-NeRF [LQC*23] reconstructs a static representation of the object using the voxel NeRF presented in [SSC22] from which particles can be sampled and evolved through a differentiable MPM simulator, managing to reduce inference timings. In both cases, the renderings produced during the optimization of the physics are compared with the ground truth for supervision. These methods provide an end-to-end pipeline for the task, but their reliance on implicit scene representations forces them to trade rendering speed for visual quality and to errors from the physical parameter estimation to appear as rendering artifacts (e.g. as gaps in the object).

Contemporaneous to our work is that of Spring-Gaus [ZYWL24]. Instead of NeRF, they utilize 3D Gaussian splatting [KKLD23] for appearance reconstruction and then sample the density field created by the Gaussian to get anchor points that can form a spring-mass system. This spring-mass system is then simulated using a differentiable engine that fits parameters such as the spring stiffness to match the images of the training set. This method is efficient in producing plausible-looking results; however, since their model is not physically grounded, the recovered parameters cannot be used to characterize the object's material.

Our method combines the explicit Gaussian splatting representation with physically based Neo-Hookean FEM simulation. In that way we combine the advantages of using an explicit representation for rendering while also being able to characterize accurately object materials. We summarize where we stand in relation to other methods in Table 1.

Method	Physically Based Parameter Estimation	Forward Simulation	Gaussian NVS
PAC-NeRF	✓	✓	
VEO	✓	✓	
Spring-Gaus		✓	✓
PhysGaussian		✓	✓
Ours	✓	✓	✓

Table 1: Summary of related methods. Our method is the only one doing physically based parameter estimation and Gaussian novel view synthesis.

3

BACKGROUND

This chapter explores the fundamental theoretical and technical background required to understand the domain of our work. We start by explaining 3D Gaussian splatting and how it can be used for novel view synthesis. We then provide some background on how elastic objects can be represented and simulated digitally and define our exact physical model.

3.1 3D GAUSSIAN SPLATTING FOR NOVEL VIEW SYNTHESIS

Gaussian splatting was initially introduced in [ZPVBG02] as a method for antialiased rendering of volumetric data. It works by using discrete 3D Gaussian kernels as rendering primitives to approximate the value of the continuous volumetric signal at every point in space. The advantage is that since the Gaussian kernels are discrete, they can be projected onto the image plane and then alpha-blended to get the final pixel values which is faster than the traditional ray marching approaches usually required for volume rendering.

Each 3D Gaussian kernel can be described by a set of parameters. Every kernel has a center position $\mathbf{c}$ [1], and a scale and rotation which can be jointly expressed in the Gaussian covariance matrix Σ. Specifically if a kernel's scale is given by the scale matrix $\mathbf{S}$ and its rotation by the rotation matrix $\mathbf{R}$, then $\mathbf{\Sigma} = \mathbf{RSS}^\mathsf{T}\mathbf{R}^\mathsf{T}$. Knowing these parameters, we can then evaluate the density of the kernel at any point $\mathbf{x}$ using the formula

$$G(\mathbf{x}) = e^{-\frac{1}{2}(\mathbf{x}-\mathbf{c})^\mathsf{T}\mathbf{\Sigma}^{-1}(\mathbf{x}-\mathbf{c})}.$$

In addition, every kernel can be given a colour, or as later proposed by Kerbl et al. [KKLD23], spherical harmonic coefficients that allow for expressing view-dependent appearance effects. With every kernel described like this we can now denote any scene as a collection of Gaussian kernels P, that combined with a view matrix $\mathbf{V}$ be used to render image I using a rendering function R.

The rendering function R can be implemented through a rasterization pipeline similar to the one used for traditional triangle-based surface rendering. As Zwicker et. al. [ZPVBG02] showed, we can define a view transformation matrix $\mathbf{M}$ that transforms the covariance matrix $\mathbf{\Sigma}$ from world space to view space ($\mathbf{\Sigma}'$). Once in view space,

1 Typically the symbol μ is used to denote the center of a Gaussian kernel. Here, we deviate from the standard notation to avoid confusion with the Lamé parameter μ, which we introduce later.

the third dimension of each Gaussian is dropped so that the kernels are projected on the image plane. Then, the colour of each pixel is calculated by alpha blending the colours of all kernels that affect it, using the kernel densities as blending weights. The whole process is summarized in Figure 4.

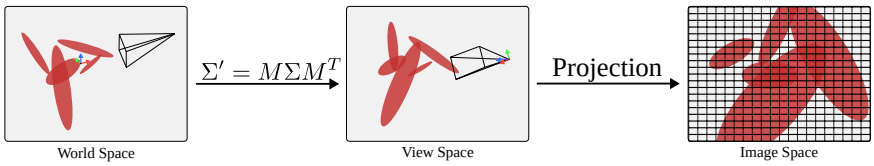


Figure 4: Overview of the Gaussian rasterization process. Similar to triangle rasterization, the Gaussians are converted from world space to view space and the projected to the image plane. The final pixel values are computed by blending the colors of each overlapping kernel.

To utilize this representation for novel view synthesis, Kerbl and colleagues [KKLD23] propose using an inverse rendering pipeline to fit the Gaussian kernels to the underlying continuous signal of the scene's radiance. The process begins by receiving a sparse point cloud along with the images of the scene and the corresponding camera poses. This initial point cloud can be randomly initialized or extracted using Structure from Motion (SfM) [SF16]. On these points, the algorithm initializes the original set of Gaussian kernels, whose position, scale, rotation, color and opacity are then optimized to fit the underlying scene.

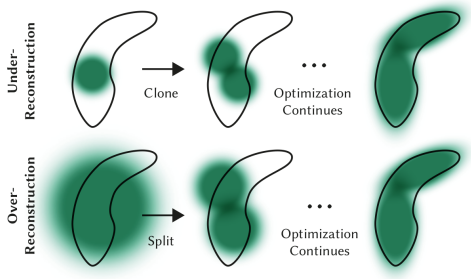


Figure 5: Adaptive Gaussian densification. The top row demonstrates that Gaussian kernels will be duplicated when the underlying geometry (black outline) is insufficiently covered. The bottom row, demonstrates that Gaussian kernels will be split in half to approximate finer geometry. Figure taken from the original 3DGS work [KKLD23]

A combination of heuristics and gradient-based optimization is used to optimize the kernels. First, the kernels are rendered using a differentiable tile-based rasterizer from all training viewpoints $\mathbf{V}_i$. The resulting image is then compared with the corresponding training set image $\hat{I}_i$ using an image loss function. Typically, this loss func-

tion is a combination of the L1 norm and structural similarity index (D-SSIM), weighted with a scalar l:

$$\mathcal{L}_{im} = (1 - l) \sum_i \|R(P, V_i) - \hat{I}_i\|_1 + l\mathcal{L}_{D-SSIM}. \quad (1)$$

Since the process is differentiable, the error gradients can be back-propagated to the Gaussian parameters, adjusting them to match input images better. On top of that, to give the algorithm the ability to create or remove Gaussians from specific parts of a scene, during the optimization step, Gaussian kernels can be split or merged depending on the error gradients to match finer geometric details (see Figure 5). The above process is repeated until the optimization converges. The entire pipeline is demonstrated in Figure 6 while Figure 7 shows the evolution of optimization for a simple scene.

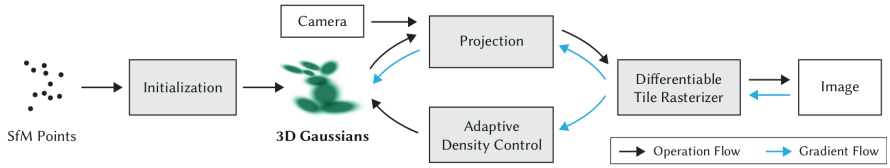


Figure 6: The Gaussian optimization pipeline. Differentiable Gaussian rasterization and densification is applied iteratively to convert an initial sparse point cloud into a 3D Gaussian representation of the underlying scene. Figure taken from the original 3DGS work [KKLD23].

Figure 7: Example Gaussian optimization process. Within a few hundred steps the Gaussians can be optimized to represent the underlying object.

The optimization process described above can achieve excellent visual quality even in large environment scenes. Concurrently, because of using rasterization (contrary to ray-marching used by NeRF-based approaches), novel views can be generated in real-time, making Gaussian splatting a good fit for interactive applications.

As a last note, we emphasize that since the Gaussian kernels are optimized in 3D space at the end of the optimization, the kernels have adapted to match the appearance and geometry of the underlying scene. Thus, if we ignore the kernels' covariance, we are left with a point cloud representation of the scene. Examples of this are shown in Figure 8.

Figure 8: Reconstructed object point cloud. By optimizing the Gaussians in 3D, the geometry of the scene is reconstruced. Here, we visualize the centers of the Gaussian kernels as points.

3.2 MODELLING AND SIMULATING ELASTIC OBJECTS

Simulating elastic objects has been an important problem in Computer Graphics since the seminal work by Terzopoulos et. al. [TPBF87]. As a result there exists a vast array of methods for this goal. In this section we try we give the minimal necessary background required to understand our work as well as the competing methods. For an in depth view of the topic we recommend the works of Kim et al. [KE20] and Nealen et al. [NMK*06].

3.2.1 *Modelling Elastic Objects*

We start by covering the basic notions of elasticity theory required for this work.

Formalizing Elasticity

In this work, we deal with hyperelastic solids that return to their original shape once all external forces are removed. This original shape is referred to as the rest shape of the object, and we can think of it as a connected continuum of points $\bar{X}$.

When the object moves and deforms, the positions of each point changes based on an affine map ϕ, resulting in a new set of points X. Note that ϕ is unique per point, so even though each point undergoes

a linear transformation, globally the object can change in a non-affine manner. For each point $\mathbf{x} \in X$ and $\bar{\mathbf{x}} \in \bar{X}$, we then have

$$\mathbf{x} = \phi(\bar{\mathbf{x}}),$$

which, because ϕ is an affine mapping, can be written as:

$$\phi(\bar{\mathbf{x}}) = \mathbf{F}\bar{\mathbf{x}} + \mathbf{t},$$

where $\mathbf{t} \in \mathbb{R}^3$ is the translation vector and $\mathbf{F} \in \mathbb{R}^{3\times3}$ is the matrix that encodes any potential rotation, scaling and reflection. The matrix $\mathbf{F}$ is referred to as the deformation gradient since it can be obtained by differentiating the previous expression:

$$\begin{aligned}\frac{\partial\phi(\bar{\mathbf{x}})}{\partial\bar{\mathbf{x}}} &= \frac{\partial}{\partial\bar{\mathbf{x}}}(\mathbf{F}\bar{\mathbf{x}} + \mathbf{t}) \\ &= \mathbf{F}\end{aligned}$$

Now that we have a way to describe the deformation that occurred, we need to define a measure of how deformed the object is. For this purpose, we use an energy function Ψ. The energy function allows us to define the "distance" of the deformed shape from the rest shape and thus can be used to determine how the object should push back against external forces to return to its rest shape.

Selecting the proper energy function is important as it affects how realistic and numerically stable the simulation will be. There are many options in the literature [KE20], but since we are building on top of DiffSim [MMG*20], we are using the Neo-Hookean energy described in [SGK18], which offers stability against element inversions and reflections, as well as maintaining the object's shape at rest and recovering from extreme compressions (e.g projecting all vertices on one line). The energy is given by the formula:

$$\Psi(\mathbf{q}, \mu, \lambda) = \frac{\mu}{2}(I_C - 3) + \frac{\lambda}{2}(J - \alpha)^2 - \frac{\mu}{2}\log(I_C + 1), \tag{2}$$

where $\mathbf{q}$ is the state vector of the object[2], μ, λ are the Lamé parameters[3], α is the rest stability term, I_C is the first Green-Cauchy invariant and $J = \det\mathbf{F}$. We explain these terms in more detail in the following paragraphs.

The first Green-Cauchy invariant, I_C, measures the deformation of an infinitesimal volumetric element of the solid. Concretely, $I_C = \sigma_x^2 + \sigma_y^2 + \sigma_z^2$, where $\sigma_x, \sigma_y, \sigma_z$ are amount of stretching of the volumetric element across each of its axis (see Figure 9). In practice I_C can be computed directly from $\mathbf{F}$ using the Frobenius norm:

$$I_C = \|\mathbf{F}\|_F^2$$

2 For example, in the case of a tetrahedral element, this state vector would be the collection of the tetrahedron's vertices

3 Technically, μ and λ as they appear in Equation 2 are shifted versions of the actual Lamé parameters. Per [SGK18] we have $\mu = \frac{4}{3}\mu_{\text{Lamé}}$ and $\lambda = \lambda_{\text{Lamé}} + \frac{5}{6}\mu_{\text{Lamé}}$. This remapping does not affect our discussion, but it is important to remember when comparing ours with other simulation techniques.

Intuitively, it measures deformation invariant to rotation (since the norm of $\mathbf{F}$ is invariant under rotation).

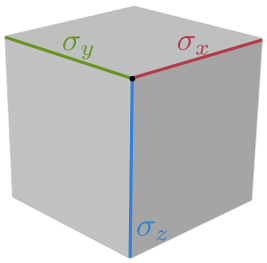


Figure 9: The components of first Green-Cauchy invariant for an infinitesimal volumetric element.

In the second term of the sum, we encounter J and α. Starting with J, we have $J = \det \mathbf{F}$, which describes how much the volume of the element has changed. It is thus essential for ensuring that the volume of the element is preserved. Additionally, we want the object to have rest stability to preserve the volume, meaning it does not collapse when it is not under any deformation. This is achieved by the α term, which is a constant set to $\alpha = 1 + \frac{\mu}{\lambda} - \frac{\mu}{4\lambda}$ (see [SGK18] for the derivation).

The last remaining terms in Equation 2 are μ and λ, which appear as scalar factors. These are the Lamé parameters and they define how much edge length and volume preservation we require for each volumetric element making up the object. Specifically, the parameter μ (also known as shear modulus) controls the length preservation while the parameter λ controls volume preservation. The two parameters form a trade-off with each other, where if we want to preserve volume, we have to set λ to be larger than μ and conversely for stronger length conservation. Last, the Lamé parameters are tightly coupled with two popular parameters describing elastic materials: Young's modulus (E) and the Poisson ratio (ν). The relations connecting them are given by:

$$E = \frac{\mu(3\lambda + 2\mu)}{\lambda + \mu}$$
$$\nu = \frac{\lambda}{2(\lambda + \mu)}$$

The E, ν pair is quite popular in literature, and thus, even though our model is parameterized directly with μ, λ, we use E and ν for our evaluations in Chapter 5.

Assumed Material Model

With the mathematical model established above, we can now describe non-linear isotropic materials. For clarity, we explain each term separately below:

- Non-Linear: Contrary to the well-known Hooke's law, we assume that the relationship between force and object displacement is not linear.
- Isotropic: We assume that the object will deform the same regardless of the direction of the force causing the deformation.
- Spatially Invariant / Homogenous: The elasticity parameter μ, λ (or equivalently E, ν) are the same across the object's surface. Note that our model does allows for varying materials, but we found the elasticity parameter optimization process to be unstable when dealing with this case, so we impose it as an extra constraint.

This material model is practical due to its simple formulation (Equation 2), while it accurately captures the behaviour of many real-life materials like rubber or numerous types of plastic [CTS*22].

3.2.2 *Simulating Elastic Objects*

In Section 3.2.1, we explain that deforming objects tend to stay as close to their best shape as possible. Further, we established how Neo-Hookean energy (see Equation 2) can measure the distance between the deformed and rest shapes. What remains is how the energy function can be used to alter the object's state such that the deformation conforms to the assumed material model.

We start this part with a general overview of time integration for simulating meshes to explain how deformation occurs in the simulation. Later, we discuss three different approaches to discretizing the simulation domain, which is essential in understanding how the material model is enforced. Specifically, we discuss the finite element method (FEM), which is the one used by our simulator, as well as two alternative approaches; the material point method and mass-spring systems.

Making Objects Move and Deform

We start with the view that objects at their rest shape can be described by a set of points X. Using the semi-implicit (symplectic) Euler integration, we can calculate how each point $\mathbf{x} \in X$ will move under the

influence of a gravitational acceleration vector $\mathbf{g}$ between two discrete time steps t and $t+1$ with Δt being the size of the time step:

$$\begin{aligned}\mathbf{u}_{t+1} &= \mathbf{u}_t + \mathbf{g}\Delta t \\ \mathbf{x}_{t+1} &= \mathbf{x}_t + \mathbf{u}_{t+1}\Delta t\end{aligned}$$

where $\mathbf{u}$ denotes the velocity vector.

External forces can be added into this formuation using Newton's second law of motion:

$$\mathbf{F}_t = m\mathbf{a}_t,$$

where $\mathbf{F}_t$ is the total force applied to the object's point at time t, m is the point's mass, and $\mathbf{a}_t$ is the point's acceleration. This equation can be rewritten to solve for acceleration:

$$\mathbf{a}_t = \frac{\mathbf{F}_t}{m}.$$

Furthermore, this acceleration can be inserted into the symplectic Euler equation for velocity in the place of $\mathbf{g}$:

$$\begin{aligned}\mathbf{u}_{t+1} &= \mathbf{u}_t + \mathbf{a}_t\Delta t \\ &= \mathbf{u}_t + \frac{\mathbf{F}_t}{m}\Delta t \quad (\text{from } \mathbf{a}_t = \frac{\mathbf{F}_t}{m}).\end{aligned} \tag{3}$$

The velocity can then be used to compute the new positions as:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{u}_{t+1}\Delta t \tag{4}$$

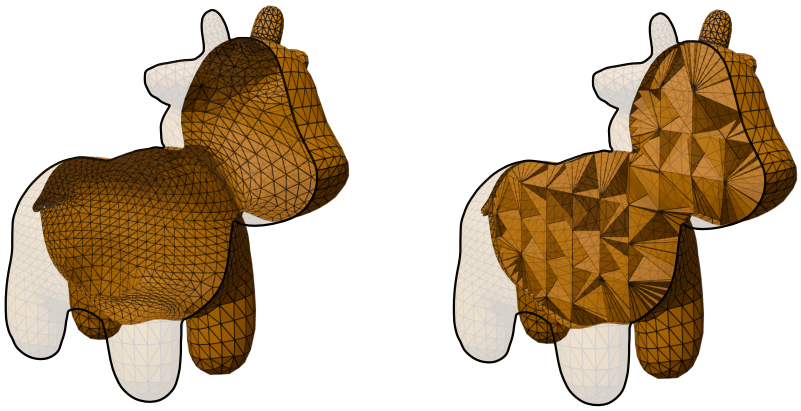
This formulation allows us to displace the object's points based on arbitrary external forces, such such as collision and elastic forces. In turn, we can deform the object based on the Neo-Hookean energy Ψ from Equation 2. Since objects deforming tend to stay as close as possible to their rest shape, we can compute the elastic force as the vector that moves the points in the direction that minimizes Ψ, which mathematically is the negative gradient of the energy function[4]:

$$\mathbf{F}_{\text{elastic}} = -\frac{\partial\Psi}{\partial\mathbf{x}_t},$$

and can be analytically derived from Equation 2 (see [SGK18]).

So far, in this analysis, we have abstractly referred to points from the object. However, in practice, we need a way to sample those points and connect them to define a continuous volume. There are multiple ways this can be done, and it depends on the way of discretizing the simulation domain. The following paragraphs discuss the three popular alternatives for simulating deformable objects.

4 Technically the gradient should also be scaled depending on the volumetric element used (e.g. inverse tetrahedron volume), but since this depends on the discretization of the simulation domain we overlook it for now.



(a) Surface triangular mesh (b) Volume tetrahedral mesh

Figure 10: Crossections of a triangular and tetrahedral mesh of the same object. In the tetrahedral case, the mesh has internal structure.

The finite element method

The finite element method (FEM) is one of the most popular approaches for simulating deformable objects. Broadly, FEM belongs to a family of simulation techniques characterized as mesh-based Lagrangian, meaning that they use connectivity information between the participating particles to do their computations. Specifically, in FEM-based simulations, the object is viewed as a continuous connected volume, which is then discretized into an irregular mesh. The specific way this discretization is done is important since it defines how the deformation gradient $\mathbf{F}$ is computed and how well the simulation mesh matches the underlying object.

In our case, we use tetrahedral meshes. Contrary to the typical triangle surface meshes used often in Computer Graphics, tetrahedral meshes are composed of tetrahedra and, as shown in Figure 10, have internal structure.

Tetrahedral-based FEM offers a simple and effective way to simulate elastic deformation since the topology of the object does not change during deforming. Additionally, simulated tetrahedral meshes can easily be used to drive the deformation of triangle meshes and (the more relevant for our case) Gaussian kernels.

The material point method

An alternative to FEM is the material point method (MPM), which has been used for our task by PAC-NeRF [LQC*23]. MPM belongs to a family of simulation techniques characterized as hybrid Eulerian-Lagrangian. This means that their computations happen both in a background grid (Eulerian) and in particles that exist and interact

with that grid (Lagrangian). Notice that no mesh is involved here, meaning no direct connectivity information exists. This lack of reliance on meshes makes MPM a good fit for simulating objects without fixed topology, such as sand or fluids. However, it can also be used for simulating hyperelastic objects by calculating the energy function (for example, the Neo-Hookean energy) on the grid and using that connectivity information to derive the forces, which can be applied to the particles. For a complete overview of MPM and its applications, we refer to [JST*16].

Mass-Spring Systems

Mass-spring systems are another instance of a mesh-based Lagrangian method for simulating deformable objects. It is a simple and intuitive approach in which any mesh can be simulated by assigning a mass at each vertex and treating each edge as a spring with stiffness k_s.

Due to their simplicity, mass-spring systems are easy to implement and computationally efficient. However, their simplicity comes at the cost of physical accuracy [NMK*06]. Specifically, the results of most such models depend on the mesh resolution and topology while increasing the mesh resolution does not necesserily lead to convergence to the true deformation state. Furthermore, the spring coefficients used are typically arbitrary and have no direct equivalency to material properties. The above makes mass-spring systems a poor fit for our purpose since our ultimate goal is to extract meaningful material properties from real-life objects and create accurate novel-interactions with them.

4

METHOD

From the related works presented in Chapter 2 and the background in Chapter 3, we recognise that there is no solution that combines physically accurate system identification with the advantages Gaussian splatting brings to novel view synthesis. Our method aims to unite these two to enable the joint optimization of appearance and physical properties with a single technique.

In the following sections, we start with an overview of the assumptions under which our method works and then proceed to explain its exact workings.

4.1 SETUP AND ASSUMPTIONS

The input to our method is multi-view video. Since we focus on single object reconstruction we require masks to be available to isolate the object. Note that those can be automatically generated through background matting [LRS*21] or semantic segmentation [KMR*23]. Moreover, we assume that on the first frame the object is fully visible and that any external forces are known. Specifically, we deal only with the case of objects free-falling under the influence of gravity and ignore air resistance[1]. We assume a non-linear isotropic and homogenous material model as explained in Section 3.2.1.

4.2 OVERVIEW

Our method consists of a two-stage reconstruction pipeline. In the first stage (left part of Figure 11), we utilize the multi-view video of the object deforming to track its deformation through an adapted version of dynamic Gaussian optimization [LKLR24]. In the second stage (right part of Figure 11), we reconstruct the object and employ a differentiable physics engine to recover the elasticity parameters, using our extracted 3D Gaussians trajectories as supervision. With this process, we can fit the parameters of our elasticity model. With the fitted parameters and the recovered mesh we can then simulate realistic novel object behaviour while also rendering the object using the Gaussians.

1 We have chosen this for simplicity, but our method can function for any known external forces.

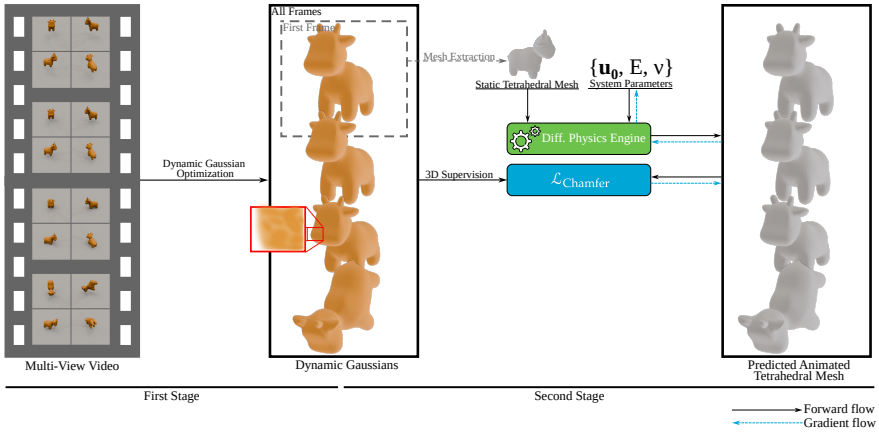


Figure 11: Overview of our appearance reconstruction and elasticity parameter estimation pipeline. In the first stage we optimize dynamic 3D Gaussian kernels based on the given multi-view video. In the second stage a tetrahedral mesh is extracted from the optimized kernels and we use their trajectories as supervision to optimize the elasticity parameters through a differentiable physics engine.

4.3 DYNAMIC GAUSSIAN OPTIMIZATION

To recover the physical parameters of the observed system, we require a form of ground truth state that the differentiable physics engine can use as supervision. We propose optimizing dynamic 3D Gaussian kernels to track how the object deforms by adapting the technique proposed in Dynamic 3D Gaussians [LKLR24]. We use the first frame frame to initialize the set of Gaussians that will be used to track the object deformations. Then, we fix the amount of Gaussians and optimize their position and size across frames. In the end we acquire a time-coherent point cloud like the one shown in Figure 12.

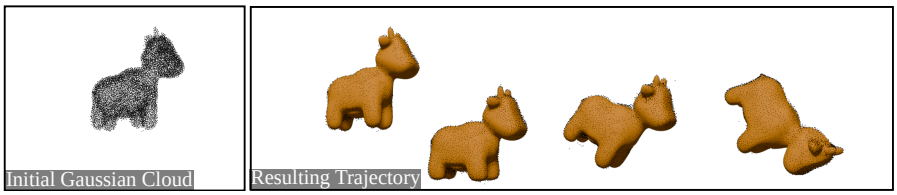


Figure 12: Extracted dynamic Gaussian kernels (visualized as points) for an example scene. By overlaying the Gaussians with the tracked object (orange cow) we see that the kernels effectively track the object throughout its deformation.

Using only the image loss $\mathcal{L}_{\text{im}}$ from Equation 1 for optimizing the positions can cause the Gaussians to move in their local neighbourhood, resulting in noisy trajectories with the Gaussians changing their position relative to the underlying object's surface. For this reason, the authors of Dynamic 3D Gaussians propose to use a local rigidity loss, which enforces rigid transformations of local Gaussian neigh-

bourhoods between consecutive frames and an isometry loss which maintains the Gaussian relative distances across all frames.

Specifically, for two Gaussians i, j with centers $\mathbf{c}_{i,t}$ and $\mathbf{c}_{j,t}$ and rotation matrices $R_{i,t}$ and $R_{j,t}$ at time t, we define the rigidity loss between them to be

$$\mathcal{L}_{i,j}^{\text{rigid}} = w_{i,j} \| (\mathbf{c}_{j,t-1} - \mathbf{c}_{i,t-1}) - R_{i,t-1} R_{i,t}^{-1} (\mathbf{c}_{j,t} - \mathbf{c}_{i,t}) \|_2^2$$

where $w_{i,j}$ is the isotropic Gaussian weighting factor:

$$w_{i,j} = e^{-\lambda_w \| \mathbf{c}_{j,0} - \mathbf{c}_{i,0} \|_2^2},$$

with $\lambda_w = 2000$ to give a standard deviation of 2.2cm. This loss is calculated in neighbourhoods of k Gaussians (determined by the knn function) and summed for the entire Gaussian cloud G:

$$\mathcal{L}^{\text{rigid}} = \frac{1}{k|G|} \sum_{i \in G} \sum_{j \in \text{knn}(i,\, k)} \mathcal{L}_{i,j}^{\text{rigid}}$$

In all our experiments we use $k = 20$. Similarly the isometry loss can be calculated as:

$$\mathcal{L}^{\text{iso}} = \frac{1}{k|G|} \sum_{i \in G} \sum_{j \in \text{knn}(i,\, k)} w_{i,j} \left| \|c_{j,0} - c_{i,0}\|_2^2 - \|c_{j,t} - c_{i,t}\|_2^2 \right|$$

We can now combine the different loss functions with weights $\boldsymbol{\omega}$ to get the total loss as:

$$\mathcal{L}_{\text{trajectories}} = \omega_0 \mathcal{L}_{\text{im}} + \omega_1 \mathcal{L}_{\text{rigid}} + \omega_2 \mathcal{L}_{\text{iso}} \tag{5}$$

Throughout our experiments we use $\omega = \{500, 1000, 700\}$. Lastly, note that contrary to the original Dynamic 3D Gaussians technique, we do not regularize the rotation of the Gaussians across frames, since in our case we care only for the quality of the resulting trajectories and not for rendering each observed frame.

Additionally, depending on the views and the object's shape, the Gaussian cloud created by optimizing the first frame can have non-uniform density, with specific areas having more Gaussians than others. Although this does not affect the reconstructed appearance, it can lead to noise arising in the optimized trajectories, as those extra Gaussians tend to cluster and move randomly inside the object. Therefore we apply Poisson subsampling to the initial Gaussian centers to enforce a blue noise distribution of the kernels. Figure 13 demonstrates the effect of this choice. Note that we still use the complete set of Gaussians for the appearance reconstruction.

4.4 MESH EXTRACTION

Concurrently, with optimizing the dynamic Gaussians, we reconstruct the object as seen in the first frame. We use the first frame Gaussian positions from the dynamic Gaussian optimization and apply

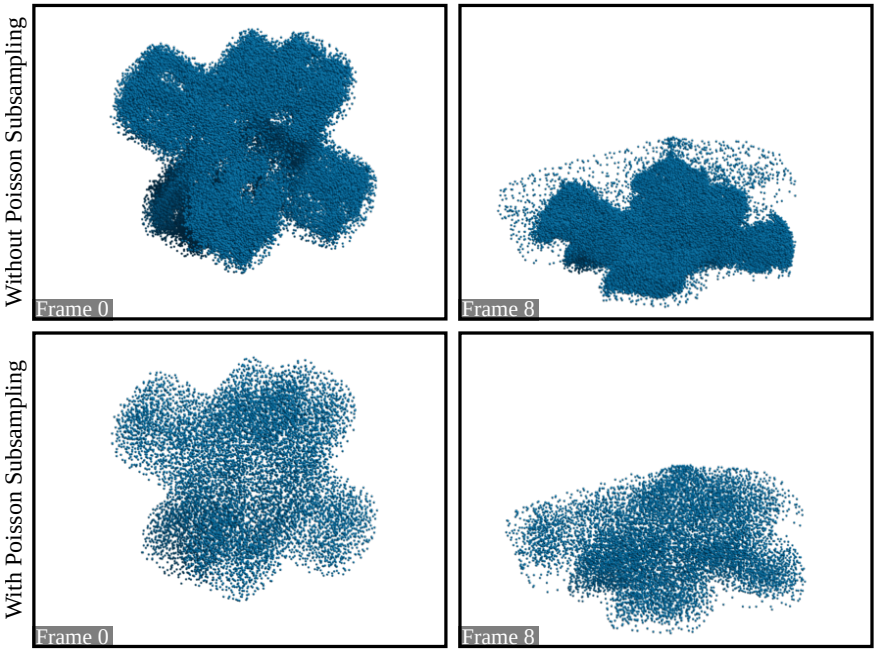


Figure 13: Impact of Poisson subsampling the Gaussian cloud. Notice that without subsampling, on frame 8 the distribution of the Gaussians is uneven inside the object, leading to noisy trajectories.

screened Poisson surface reconstruction [KH13]. Since the Gaussian cloud is not perfectly aligned with the surface and can have internal points the resulting mesh is often of low-quality and thus it is ill-suited for simulation (see Poisson reconstructed mesh in Figure 14). To mitigate this issue, we apply alpha wrapping [PRLH*22] which

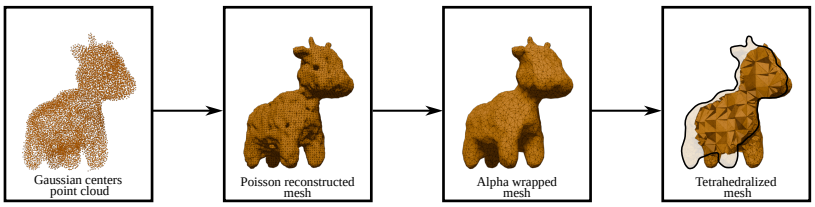


Figure 14: The stages of our mesh extraction process. We rely on multiple methods from traditional geometry processing to reconstruct the object from the Gaussians.

computes a new enclosing mesh of higher quality (see alpha wrapped mesh in Figure 14). At this point we have recovered a surface mesh, but as explained in Section 3.2 we require the mesh to have internal structure so that it can be simulated. To obtain the internal structure we tetrahedralize the results of alpha wrapping using Delaunay tetrahedralization. Delaunay tetrahedralization populates the inside volume by placing points inside the surface, and connects them in a way that avoids creating slim or degenerate tetrahedra and maintains

their uniformity. As a result, the output tetrahedral mesh is suitable for FEM based simulations. In the end we obtain the final simulatable mesh (X_0, A), where X_0 is the set of vertices and A is the set of edges (adjacencies).

4.5 PHYSICAL PARAMETER RECOVERY

Once we have both the tetrahedral mesh and the dynamic Gaussian cloud, we can recover the initial velocity of the object and the elasticity parameters describing its material. As a reminder, we are following the Neo-Hookean elasticity model (Equation 2) for homogenous materials presented in Section 3.2.1, and thus we need to optimize two parameters; Young's modulus E and the Poisson ratio ν.

To recover the initial velocity $\hat{\mathbf{u}}$ and the optimal elasticity parameters $\hat{E}, \hat{\nu}$, we employ the differentiable physics engine DiffSim [MMG*20]. The physics engine can produce the deformed mesh vertex positions X_t for every moment t based on a given initial velocity $\mathbf{u}_0$ and elasticity parameters E, ν. Formally, we denote the engine as a function

$$S : (\mathbf{u}_0, E, \nu, X_0, A, t) \rightarrow X_t,$$

that evolves the system state using the rules of Equation 3 and Equation 4.

We can now use this formulation to define a loss term that can be used to perform gradient-based optimization on the parameters. Since we have no reliable way to get exact correspondences between the mesh's vertices and the dynamic Gaussians, we use the Chamfer distance between the Gaussian centers and the vertices:

$$\mathcal{L}_{\text{Chamfer}}(X_t, G_t) = \sum_{\mathbf{x} \in X_t} \min_{\mathbf{c} \in G_t} \|\mathbf{x} - \mathbf{c}\|_2^2 + \sum_{\mathbf{c} \in G_t} \min_{\mathbf{x} \in X_t} \|\mathbf{x} - \mathbf{c}\|_2^2,$$

where G_t is the set of dynamic Gaussian centers at time moment t.

To make the optimization more robust, we follow a scheme similar to that described in PAC-NeRF [LQC*23]. We initialize the elasticity parameters at values E_0, ν_0 and use the states observed during free-fall to optimize only the velocity. This approach is effective because those states provide a clear learning signal for the velocity. This way, we also ensure that the elasticity parameters are not adjusted to fix mismatches in velocity. Formally, if contact with the ground occurs at time moment N_c, we have:

$$\hat{\mathbf{u}} = \text{argmin}_{\mathbf{u}} \frac{1}{N_c} \sum_{t=0}^{N_c - 1} \mathcal{L}_{\text{Chamfer}}(S(\mathbf{u}, E_0, \nu_0, X_0, A, t), G_t).$$

After the velocity is optimized, we fix it and proceed to fit E and ν, which are optimized using the entire sequence of N observed states:

$$\hat{E}, \hat{\nu} = \text{argmin}_{E,\nu} \frac{1}{N} \sum_{t=0}^{N-1} \mathcal{L}_{\text{Chamfer}}(S(\hat{\mathbf{u}}, E, \nu, X_0, A, t), G_t).$$

In the end, the above optimization process allows us to recover the set of parameters $\hat{\mathbf{u}}$, $\hat{\mathbf{E}}$ and $\hat{\mathbf{v}}$, that explain the observed dynamics.

4.6 GENERATING NOVEL VIEWS AND DYNAMICS

Given new positions for the vertices of the mesh $\mathbf{X}_0'$ and a new initial velocity $\mathbf{u}'$, we simulate novel dynamics using $S(\mathbf{u}', \hat{\mathbf{E}}, \hat{\mathbf{v}}, \mathbf{X}_0', A, t)$. We propagate the resulting mesh motion to our original Gaussian representation by updating the kernel positions, scales and rotations.

To change the kernels' positions, we use Mean Value Coordinates (MVC) [JSW23]. This is different to previous approaches that have used barycentric coordinates for the same purpose [ZBS*23; JYX*24], and we evaluate our choice in Section 5.10.

For adjusting the scale and rotation we assign each Gaussian to the closest tetrahedron and apply deformation transfer [SP04; ZBS*23]. Concretely, for a tetrahedron i we define the deformation matrix $\mathbf{J}_i$ as:

$$\begin{aligned}\mathbf{J}_i\mathbf{E}_i &= \hat{\mathbf{E}}_i \\ \mathbf{J}_i &= \hat{\mathbf{E}}_i\mathbf{E}_i^{-1},\end{aligned}$$

where $\mathbf{E}_i \in \mathbb{R}^{3\times3}$ and $\hat{\mathbf{E}}_i \in \mathbb{R}^{3\times3}$ contain the edge vectors of tetrahedron i in the canonical and deformed states. We provide a visual explanation of this formula in Figure 15. The deformation matrix is

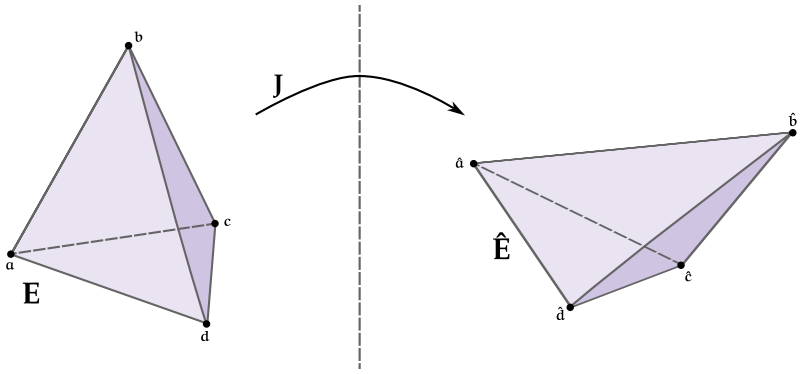


Figure 15: Explanation of $\mathbf{J}$. Given a tetrahedron with vertices $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$, we descibe its initial state as $\mathbf{E} = [\mathbf{b}-\mathbf{a}, \mathbf{c}-\mathbf{a}, \mathbf{d}-\mathbf{a}]$ and its deformed state as $\hat{\mathbf{E}}$ (computed in a similar manner). With these, we can now define $\mathbf{J}$ as the transformation from $\mathbf{E}$ to $\hat{\mathbf{E}}$, or equivalently: $\hat{\mathbf{E}} = \mathbf{J}\mathbf{E}$.

then applied directly to adjust the Gaussian covariance matrix:

$$\hat{\mathbf{\Sigma}}' = \mathbf{J}_i\mathbf{\Sigma}\mathbf{J}_i^T.$$

As with MVC, we demonstrate the importance of this in our ablations in Section 5.11.

With MVC and the tetrahedral deformation transfer, we are able to use the simulated mesh to guide the Gaussian kernels. In this way our method allows for rendering novel views of novel object interactions.

5 EXPERIMENTS

In this chapter, we evaluate our method through a series of experiments. We start by providing some implementation details about our method and describing the baselines and datasets we used. We then examine our method's physical parameter estimation and appearance reconstruction capabilities and provide timings for our method's inference pass. We then ablate the different steps of our reconstruction pipeline and examine the impact of the mesh reconstruction on our dynamics reconstruction. From there, we conduct experiments to chart the optimization landscape of our method. Last, we experimentally demonstrate the importance of using MVC and tetrahedral deformation transfer to adjust the Gaussian kernels.

5.1 IMPLEMENTATION DETAILS

We implement our method in PyTorch [PGC*17] using the custom CUDA backends for differentiable Gaussian rasterization [KKLD23] and differentiable physics simulation [MMG*20]. For the mesh processing used in our mesh extraction step, we use the corresponding MeshLab [CCC*08] filters with their default parameters. Simultaneously, we rely on TetGen [Han15] for the Delaunay tetrahedralization. Last, we use the MVC implementation of Bergman [BKY*22a].

We initialize the Gaussians with a random point cloud and use 10,000 iterations to optimize the initial frame and then subsample the cloud to 12000 Gaussians. After, we use 3500 iterations to optimize the dynamics. For simplicity, we optimize colour and not spherical harmonics throughout the Gaussian optimization. In the image loss (Equation 1) we use $l = 0.2$. Regarding inverse physics, we use 12,000 simulation substeps in our dataset and 10,000 for the Elasticity one with 100 iterations to optimize the velocity and 120 iterations to optimize the elasticity parameters. Across the board, we use the Adam optimizer [KB15]. The learning rates for each parameter are shown in Table 2 and are consistent across scenes. We use a machine with an NVIDIA GeForce RTX 4070 Ti SUPER with 16GB of GPU memory for Gaussian and inverse physics optimization. For training PAC-NeRF, we use a single Tesla V100 GPU with 32GB of memory offered by the DelftBlue HPC cluster [DHP24] since PAC-NeRF's training has higher GPU memory requirements. We time the inference passes of both techniques in the machine with the RTX 4070.

Parameter	Learning Rate
Gaussian Centers	0.0005
Gaussian Colors	0.0025
Gaussian Rotations	0.001
Gaussian (logit) Opacity	0.05
Gaussian (log) Scales	0.001
$\mathfrak{u}$	0.001
(log) E	0.1
ν	0.001

Table 2: Summary of learning rates used for our optimizations.

5.2 BASELINES

To the best of our knowledge the only other techniques that do joint physical parameter estimation and novel view synthesis are VEO [CTS*22] and PAC-NeRF [LQC*23] (see Table 1). However, VEO does not openly provide an implementation, making it difficult to compare against them in this work's timeframe. For this reason, we evaluate our method only against PAC-NeRF. Note that the authors of PAC-NeRF compared their technique against VEO by re-implementing the latter and found that their approach outperformed VEO significantly in all aspects. Thus we assume that it is sufficient to compare only with PAC-NeRF.

5.3 DATASETS

For our evaluation, we rely on two synthetic datasets. The first one is the Elasticity dataset, created by the authors of PAC-NeRF using their simulator. It consists of 10 scenes of the same object falling with different elasticity parameters and initial configurations. Although the Elasticity dataset is sufficient to evaluate physical parameter recovery, it relies only on one simple monochrome object. Thus, it is not well suited for evaluating the performance of the appearance reconstruction. For this reason, we also create our own dataset of three scenes, each with a different object and simulation parameters. We simulate the objects using DiffSim and render the results in Blender [Com18]. To avoid any biases caused by the camera views used, we use the same ones as in the Elasticity dataset. Overviews of the two datasets are shown in Figure 16 and in Figure 17.

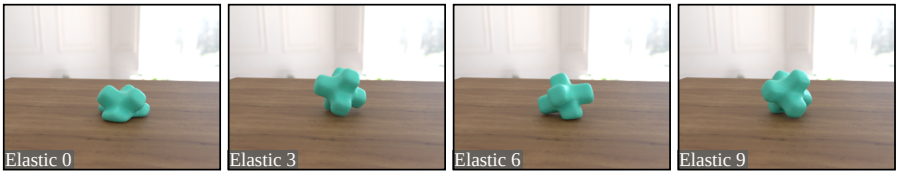


Figure 16: Frame 5 from selected scenes of the Elasticity dataset. Every scene consists of the same object with different initial configuration and elasticity parameters. Here we show the same time moment from the same camera view for 4 different scenes (out of the total 10).

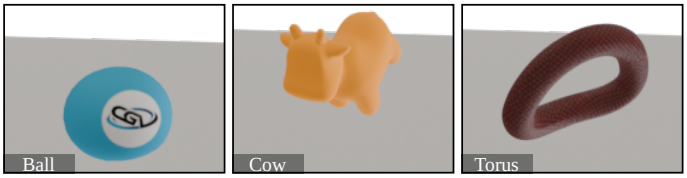


Figure 17: Our dataset. We use the Torus and Ball scene to evaluate appearance reconstruction of complex textures and the Cow scene to evaluate dynamic and appearance reconstruction of more complex geometry.

5.4 PHYSICAL PARAMETER ESTIMATION

We compare our method to PAC-NeRF [LQC*23] across physical parameter estimation performance. Both techniques recover the initial velocity with high accuracy ($\approx$ 1% average error), so we do not report the detailed results, while Table 3 show the elasticity parameter estimation results for each scene.

Across both datasets, the performance of our technique is often better than that of PAC-NeRF. However, we note that each technique performs better than the other on the dataset created with the same physics simulator. This discrepancy implies that both techniques are biased to perform better on data that matches their physical model. Nonetheless, our method can produce accurate predictions for both datasets, often outperforming PAC-NeRF in one or both parameter estimates. This is not true of PAC-NeRF, which consistently exhibits worse performance in our dataset.

Interestingly, we note that for both techniques, there are scenes for which performance drops significantly in one or both parameter estimates. We examine possible explanations of this behaviour in Section 5.9.

5.5 APPEARANCE RECONSTRUCTION

We now evaluate the performance of our method in appearance reconstruction. As a reminder, we perform 3D Gaussian optimization only on the first frame and use the underlying reconstructed dynamics to move, scale and rotate the Gaussians accordingly. We use the

	$\log_{10}(E)$ (↓)		ν (↓)	
	PAC-NeRF	Ours	PAC-NeRF	Ours
Elastic 0	0.02	0.262	<0.001	0.038
Elastic 1	**0.04**	0.06	0.03	**0.014**
Elastic 2	**0.008**	0.185	**0.069**	0.1
Elastic 3	**0.019**	0.049	0.04	**0.01**
Elastic 4	0.002	0.213	0.017	0.067
Elastic 5	0.088	**0.043**	**0.075**	0.118
Elastic 6	**0.031**	0.167	<0.001	0.060
Elastic 7	0.149	**0.028**	**0.077**	0.140
Elastic 8	**0.113**	0.288	0.261	**0.135**
Elastic 9	0.031	0.195	0.055	0.130
Ball	0.154	**0.03**	0.2789	**0.039**
Cow	0.150	**0.011**	0.0581	**0.026**
Torus	1.036	**0.062**	0.0833	**0.005**
Mean	0.141	**0.122**	0.080	**0.068**
STD	0.274	0.098	0.088	0.050

Table 3: Parameter estimation results across all scenes. Our method is competitive to PAC-NeRF.

PSNR, SSIM and LPIPS [ZIE*18] image metrics and evaluate the performance of our method in the first frame (Table 4) and across the entire sequence (Table 5) for both datasets. Figure 18 and Figure 19 show some example results.

Table 4 shows that our method consistently achieves better visual quality than PAC-NeRF in static scene reconstruction. We demonstate this qualitatively in Figure 18. However, in Table 5 we see that this is not always the case when reconstructing the entire sequence. Specifically, we observe that PAC-NeRF often achieves better scores in the PSNR and SSIM metrics, even in scenes where our method has performed better in the parameter recovery (e.g. in the Ball scene). Upon inspecting the results, we find that this is because of mismatches in the reconstructed object's position and exact deformed shape. We analyse this further in Section 5.8, where we show that these mismatches arise due to the quality of the reconstructed mesh. Last, observe that even for these cases, our LPIPS score is often better than PAC-NeRF's, indicating that the mismatches are less perceivable than PAC-NeRF's rendering artifacts.

As shown in Figure 18 and Figure 20, our method can reconstruct complex textures and maintain their quality under deformation ac-

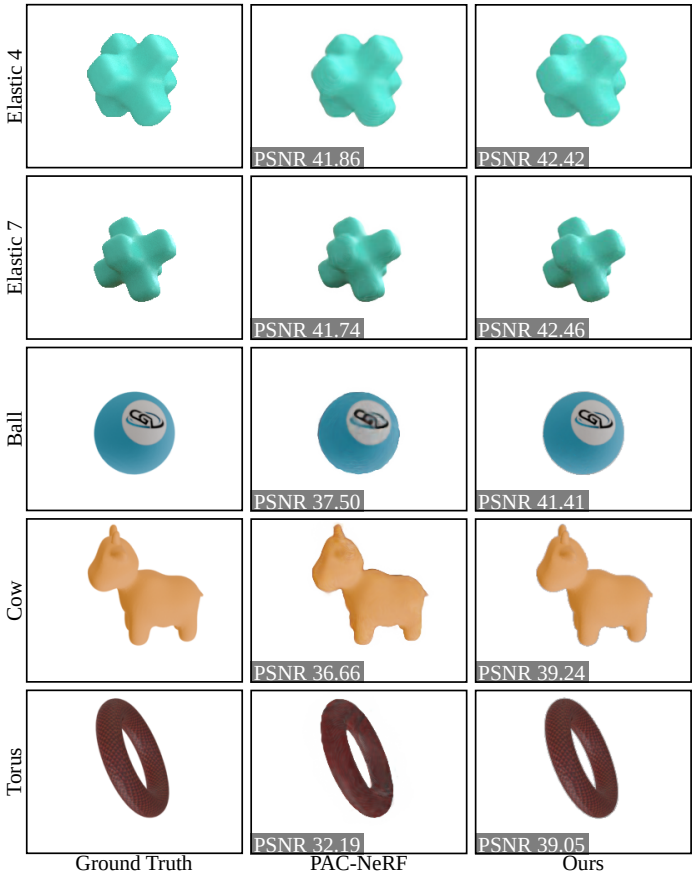


Figure 18: Static reconstruction results. Our method can accurately reconstruct uniform color objects and objects with high frequency textures.

curately. Simultaneously, we find that PAC-NeRF suffers from discretization artifacts due to the discrete voxel NeRF it uses. Our method does not have this problem since it relies on Gaussian splatting. Note that we do observe certain artifacts in our method, which we discuss in Section 6.2.4.

5.6 TIMINGS

We now discuss the performance of our method in terms of speed. Overall, our backwards pass requires 2.1 hours on average, with the main bottleneck being the inverse physics optimization ($\approx$ 1.5 hours) and the rest being required for the trajectories optimization. In the inferences, our method required on average 3.3 seconds per scene, with 96% of that allocated to the physics simulation. We summarize our method's inference performance in Figure 21. Notice that our method performs at near real-time rates and outperforms PAC-NeRF by 7$\times$ on average. In addition, according to the timings reported in VEO [CTS*22], our method is 700$\times$ faster than theirs. Last, our method

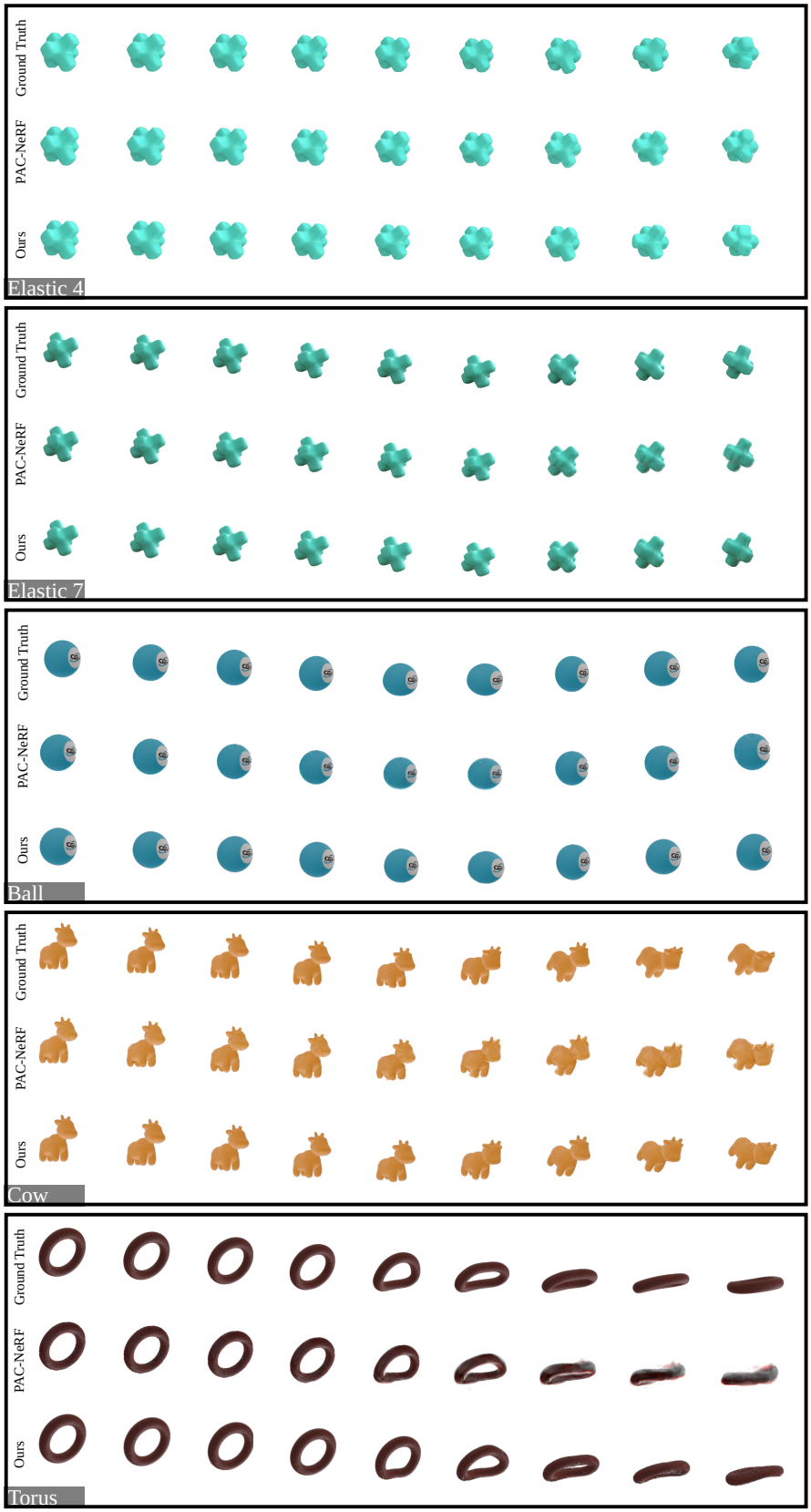


Figure 19: Dynamic reconstruction results. We demonstrate our method's ability to reconstruct observed dynamics based on the estimated elasticity parameters.

	PSNR (↑)		SSIM (↑)		LPIPS (↓)	
	PAC-NeRF	Ours	PAC-NeRF	Ours	PAC-NeRF	Ours
Elastic 0	40.89	**42.28**	0.994	**0.9945**	0.0053	**0.004**
Elastic 1	41.63	**42.19**	0.9944	**0.9945**	0.0046	**0.0035**
Elastic 2	41.33	**42.22**	0.994	**0.9944**	0.0056	**0.0042**
Elastic 3	40.77	**42.10**	0.9938	**0.9946**	0.006	**0.0043**
Elastic 4	41.86	**42.42**	0.9943	**0.9946**	0.0051	**0.0039**
Elastic 5	41.69	**42.39**	0.9941	**0.9945**	0.0052	**0.0041**
Elastic 6	41.86	**42.40**	0.9943	**0.9945**	0.0051	**0.0039**
Elastic 7	41.74	**42.46**	0.9944	**0.9945**	0.0051	**0.0039**
Elastic 8	41.86	**42.28**	0.9945	**0.9946**	0.0048	**0.0038**
Elastic 9	41.34	**42.2**	0.9941	**0.9945**	0.0051	**0.0038**
Ball	37.50	**41.41**	0.9931	**0.9967**	0.0057	**0.0039**
Cow	36.66	**39.24**	0.9904	**0.9934**	0.0031	**0.0018**
Torus	32.19	**39.05**	0.9754	**0.9964**	0.0082	**0.0056**

Table 4: Static appearance reconstruction results. Our method is consistently better than PAC-NeRF on static reconstruction.

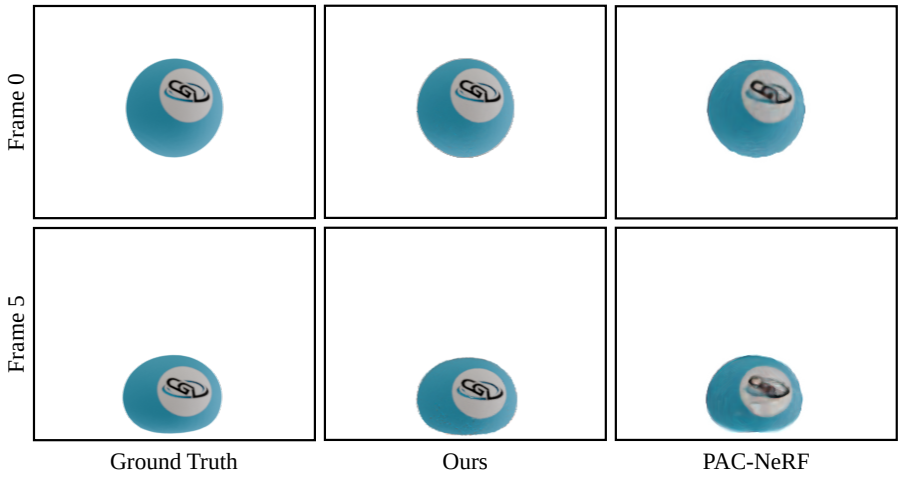


Figure 20: Qualititative evaluation on the Ball scene. Our method can reconstruct complex textures and maintain their quality under object deformation.

can be easily adjusted to use a faster physics engine if one becomes available, potentially reducing simulation times significantly.

Note that we observe differences in the simulation time of the same amount of frames. The cause for this is, first, the differences in the number of vertices and tetrahedra on the reconstructed meshes across different scenes and, second, the fact a different number of simulation substeps is required per scene. This is a consequence of the

	PSNR (↑)		SSIM (↑)		LPIPS (↓)	
	PAC-NeRF	Ours	PAC-NeRF	Ours	PAC-NeRF	Ours
Elastic 0	**34.96**	28.29	**0.9862**	0.9728	**0.0111**	0.0191
Elastic 1	**35.11**	32.59	**0.9862**	0.9804	**0.0106**	0.0121
Elastic 2	**35.99**	34.46	**0.9888**	0.9851	0.0086	**0.0085**
Elastic 3	**33.80**	24.47	**0.983**	0.9617	**0.014**	0.0424
Elastic 4	34.39	**34.70**	0.9849	**0.9859**	0.0125	**0.0089**
Elastic 5	**35.82**	33.50	**0.9878**	0.9829	**0.0108**	0.0111
Elastic 6	34.91	**35.50**	0.9855	**0.9867**	0.0118	**0.008**
Elastic 7	33.58	**33.72**	0.9825	**0.9826**	0.0137	**0.0111**
Elastic 8	**34.44**	31.29	**0.9841**	0.9764	**0.0125**	0.0218
Elastic 9	**34.10**	34.07	0.9826	**0.9829**	0.0137	**0.0105**
Ball	**33.15**	30.55	**0.9872**	0.9822	0.0057	**0.0039**
Cow	**32.44**	32.42	**0.9808**	0.9797	0.0076	**0.0073**
Torus	25.13	**25.17**	0.9558	**0.9564**	0.0219	**0.0156**

Table 5: Dynamic appearance reconstruction results. Our method is often better than PAC-NeRF. In Section 5.8, we demonstrate that these errors arise due to a mismatch in dynamics and not because of visual quality.

Neo-Hookean FEM model we use, in which the mesh and the exact parameters used affect the number of simulation substeps required for the symplectic Euler integration to remain stable.

5.7 PIPELINE STAGES ABLATION FOR PHYSICAL PARAMETER RECOVERY

We now ablate the different parts of our pipeline. To do this, we perform three experiments. First, we replace the extracted Gaussian trajectories with the ground truth vertex positions that generated the dataset. Second, we again use the Gaussian trajectories but replace the reconstructed mesh with the ground truth one. Last, we use both the ground truth mesh and trajectories to get a baseline on the physical parameter estimation. We conduct these experiment only on our dataset, since it is the only one for which we have access to the ground truths. The results of our experiments are shown in Table 6 and Figure 22.

Across our ablations and the results of Section 5.4, we see no significant difference in parameter performance and find that no version is universally better. Interestingly, we observe that in certain cases using the ground truth trajectories or mesh does not lead to better re-

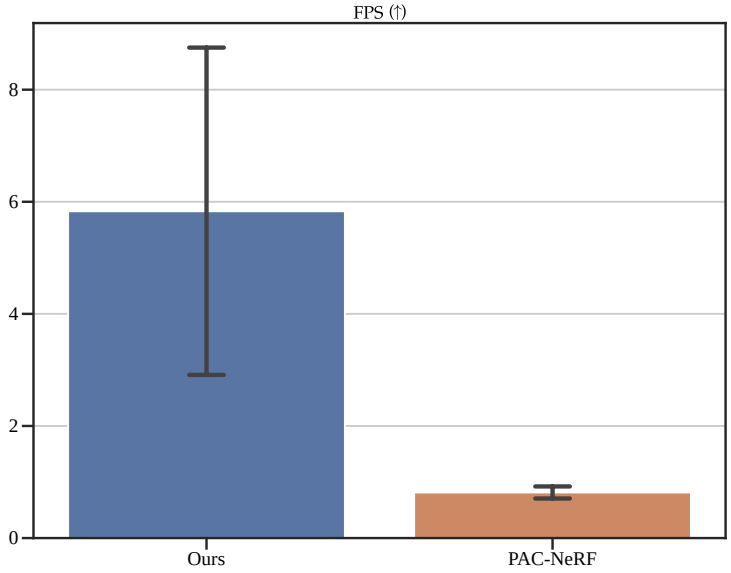


Figure 21: Perfomance in FPS of our method and PAC-NeRF. On average, our method is 7× faster than PAC-NeRF, although it displays variability in the timings due to different mesh sizes and required simulation substeps per scene.

sults. We therefore reason that the trajectories extracted and the mesh reconstruction are not the main causes of errors in the parameter estimation. We further explore this idea Section 5.9.

5.8 MESH IMPACT ON DYNAMICS

In Section 5.5, we showed examples of scenes where our method outperformed PAC-NeRF in parameter estimation but was still unable to reproduce the exact observed deformation. We now demonstrate that the root of this problem is the reconstructed mesh. For this purpose,

	$\log_{10}(E)$ (↓)	ν (↓)
Full Method	0.0346	0.0239
GT Mesh	0.2208	0.0020
GT Traj.	0.6755	0.0090
GT Mesh and Traj.	0.3248	0.0413

Table 6: We ablate our method using the ground truth (GT) mesh and GT trajectories to evaluate the impact of each pipeline stage to the physical parameter estimation. We find no configuration that is universally better. Results are averaged across all scenes of our dataset.

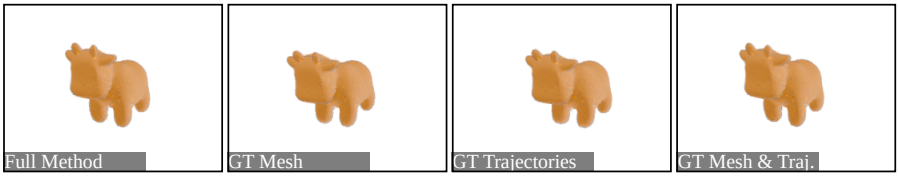


Figure 22: Example results for the ablation configurations tested. Although minor differences exist, we find that they are not immediately noticable.

we simulate the Ball scene using the exact ground truth physical parameters with the reconstructed mesh and render the deformation like before. Additionally, we use the ground truth mesh and simulate it with the recovered parameters. The results can be seen in Figure 23. We see that in the case of our method, there is a shape mismatch when using the reconstructed mesh, even though we have used the correct parameters for the simulation. We discuss this issue more and propose potential solutions in Section 6.2.6.

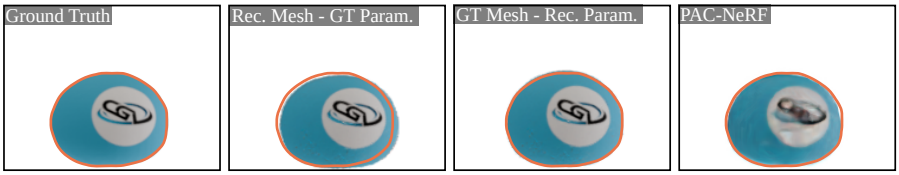


Figure 23: Mesh's impact to reconstructed dynamics. Across all images the orange border represents the ground truth shape and position. Observe that the reconstructed mesh, simulated with the correct parameters ("Rec. Mesh - GT Param."), has a bigger mismatch with the ground truth shape than the ground truth mesh simulated with the recovered parameters ("GT Mesh - Rec. Param.").

Last, we emphasize that although the mesh impacts the simulated dynamics, the ablation in Section 5.7 showed that it does not significantly affect the estimation of the physical parameter values.

5.9 UNDERSTANDING THE OPTIMIZATION LANDSCAPE

In Section 5.4, we observed outliers in the performance of the physical parameter estimation. Moreover, in Section 5.7, we showed that the errors in parameter estimation persist when we remove any noise incurred by the extracted Gaussian trajectories and mesh reconstruction. We now explore and justify these observations through experiments that map our method's optimization landscape. To do so, we perform a grid search over the parameter space and plot the resulting loss landscape. We evaluate only the differentiable physics engine here, so we use the ground truth mesh and the ground truth deforming vertices as supervision. We demonstrate the results for the Ball

scene in Figure 24, but the observed behaviours were similar to those in the other scenes of our dataset.

Interestingly, the first column of Figure 24 demonstrates that there is no single unique minimum but rather a region over μ and λ[1] for which the loss function has low values. This explains why, even in the presence of errors in the value of the elasticity parameters, the observed dynamics are still matched. This implies that the problem is under-constrained for the given training sequences. Or equivalently, the correct values of the elasticity parameters are ambiguous from the given training sequence. To further test this, we increase the number of frames used for training. The results are shown in the second and third parts of Figure 24. Observe that adding frames makes the optimization landscape less smooth and steeper, meaning that gradient descent optimization will more quickly be guided towards low loss value regions. Simultaneously, as more timesteps are used for supervision a unique minimum seems to arise. We conclude that observing more interactions is crucial for recovering the correct values of the elasticity parameters.

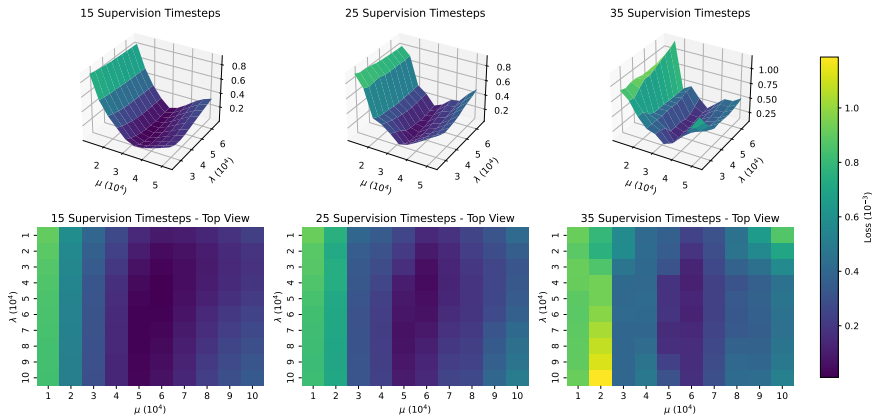


Figure 24: The optimization landscapes of the Ball scene for different amount of supervision timesteps. The top row shows the optimization landscapes in 3D while the bottom row demonstrates a 2D view of them from the top. Notice that as more timesteps are used for supervision, the landscape changes to have a single well defined minimum.

To explore this phenomenon further, we construct a scenario where the ambiguity about the parameter values should be minimal using the setup shown in Figure 25. We reason that ambiguity in this setup will be minimal because it directly corresponds with the definition of the elasticity parameters. We let an elastic beam hang and apply an initial velocity to the vertices of its bottom.

1 As a reminder μ and λ are related to E and ν (see Section 3.2.1). We use μ and λ here since DiffSim is directly parameterized with those.

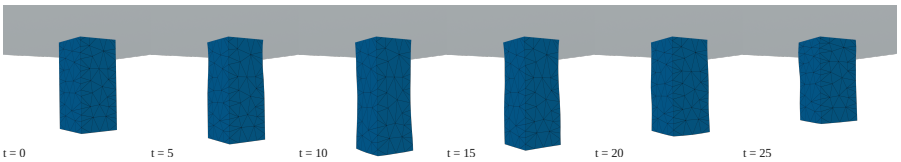


Figure 25: The Beam setup and resulting sequence. We let an elastic beam hang and set it in motion by applying some initial velocity to its bottom.

Like before, we plot the optimization landscapes using different numbers of frames for supervision. Figure 26 demonstrates the results. In this case, we observe similar behaviour to before, only now the difference is less pronounced between supervising with 25 and 35 frames. These observations imply that depending on the interactions that are being reconstructed, a different amount of training data is required for good results.

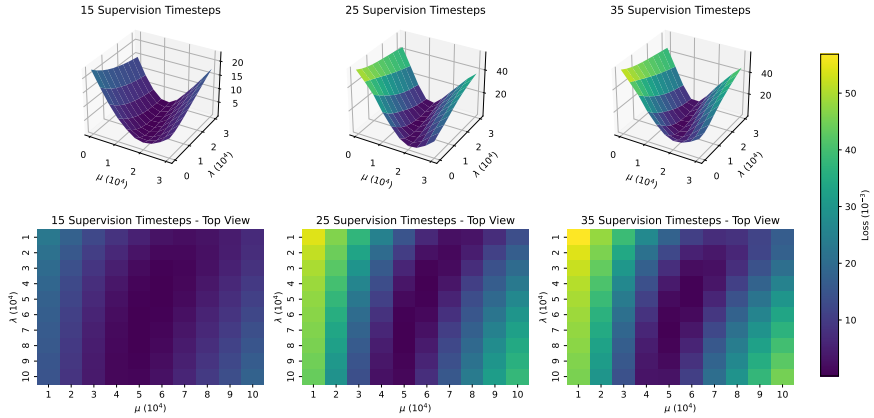


Figure 26: The optimization landscapes of the Beam scene for different amount of supervision timesteps. The top row shows the optimization landscapes in 3D while the bottom row demonstrates a 2D view of them from the top. Notice that as more timesteps are used for supervision, the landscape changes to have a single well defined minimum.

Overall, the above analysis indicates that the errors in physical parameter estimation reported in Section 5.4 can be attributed to the training set's ambiguity and not on the method.

5.10 IMPORTANCE OF MEAN VALUE COORDINATES

We now evaluate our choice of using Mean Value Coordinates (MVC) for controlling the Gaussians' positions in Section 4.6. There, we mentioned that a popular alternative is using barycentric coordinates for the same goal. In Figure 27, we demonstrate why barycentric coordinates are problematic in our case.

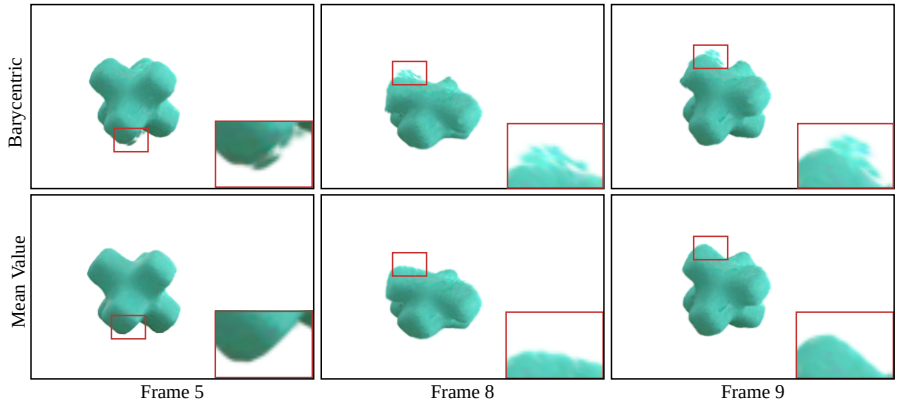


Figure 27: Comparison between using barycentric (top) and mean value (bottom) coordinates to drive the kernels' positions. In contrast to MVC, barycentric coordinates can lead to kernels moving away from the object.

In the case of barycentric coordinates, some kernels move away from the object when it undergoes extreme deformation. This happens because the mesh extracted with the Poisson reconstruction is not guaranteed to enclose all kernels. These kernels will then have negative barycentric coordinates, meaning that if the tetrahedron they belong to expands, their distance from it grows, creating the observed artifacts. This effect is demonstrated for the 2D case in Figure 28.

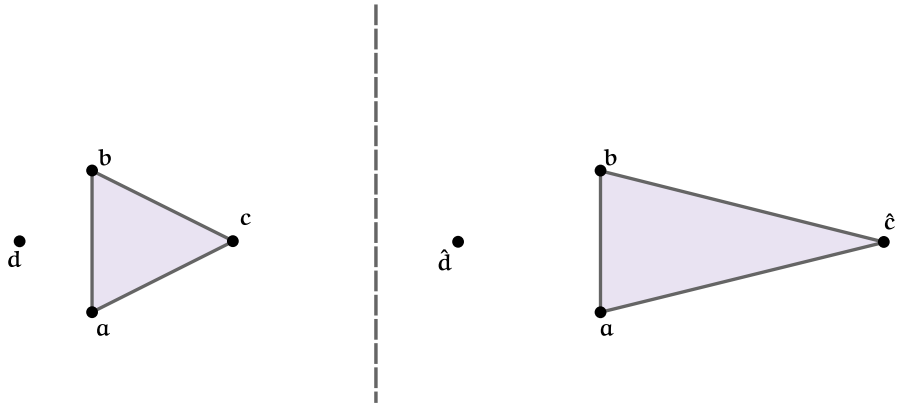


Figure 28: Cause of barycentric coordinate artifacts. In this configuration a point $\mathbf{d}$ is defined in terms of the triangle vertices $\mathbf{a}, \mathbf{b}, \mathbf{c}$. Then, as $\mathbf{c}$ deforms to $\hat{\mathbf{c}}$, the point is pushed away from the triangle.

In contrast, the behaviour of MVC for kernels outside of the cage is smoother because the MVC coordinates depend on all of the surface vertices. As a result, even if some parts of the mesh deform highly, they have a limited impact on the kernels' positions, effectively resolving the artifacts observed in the case of barycentric coordinates.

5.11 IMPORTANCE OF KERNEL COVARIANCE ADJUSTING

In Section 4.6, we demonstrated how the deformation of the cage can be used to adjust the scale and rotation (covariance) of the Gaussian kernels. We now demonstrate the effectiveness of this approach in Figure 29.

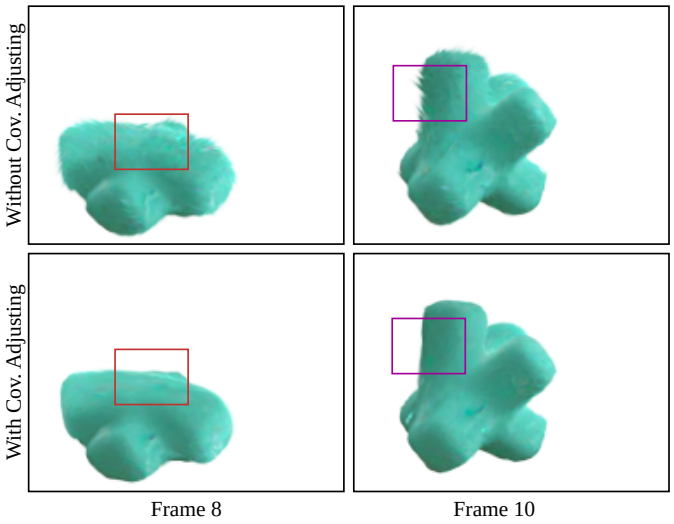


Figure 29: Impact of adjusting the kernel covariances based on the underlying mesh deformation. The colored boxes indicate areas where the difference is most pronounced.

In the top row of Figure 29, we disable adjusting the covariances and only change the positions of the kernels using MVC. Here, kernels can be observed as spiking artifacts that appear almost perpendicular to the underlying surface. These artifacts arise because the rotation of the kernels has been optimized only on the first frame, but throughout the deformation, the relative rotation of the underlying surface changes. In contrast, the bottom row of Figure 29 shows that this issue is significantly reduced when we adjust the kernels' covariance matrix based on the cage's deformation.

6 DISCUSSION AND CONCLUSION

We now discuss the results obtained in Chapter 5 and derive conclusions about our work. We further give recommendations for future work.

6.1 RESULTS

We summarise the results presented in Chapter 5.

PHYSICAL PARAMETER ESTIMATION. Our method can estimate physical parameters with precision competitive to the state-of-the-art. However, we find discrepancies in the performance of the tested methods across different datasets and thus believe their generalizability to real-world data should be studied further.

APPEARANCE RECONSTRUCTION. We find that our method successfully reconstructs object appearance, even in cases of objects with complicated textures.

TIMINGS. Our method can produce novel views and interactions at near real-time rates ($\approx$ 6 FPS on average). This is a 7$\times$ increase in the speed of the primary competing method.

6.2 LIMITATIONS AND FUTURE WORK

We now underline the limitations of our method and indicate directions for future research to address them.

6.2.1 *Setup Assumptions*

Our method assumes non-linear isotropic and homogeneous materials. As a result, it cannot reconstruct objects whose material properties change across their surface. A naive solution to this would be to optimize elasticity parameters per vertex of the mesh; however, some early experiments we conducted indicated that this makes the optimization process unstable because the recovered materials can vary between neighbouring vertices. Therefore, we suggest exploring regularization methods to constrain the variability of the material across the surface to the minimum required.

Moreover, our method assumes that multi-view video is available, which can often be challenging to acquire. Exploring the applicability

of monocular reconstruction methods [TKB*23] can provide ways to remove this requirement.

Additionally, our method requires object masks to be available for every frame in the input. Although, high quality masks can be generated automatically [LRS*21; KMR*23], our method can potentially be adapted to work without them by performing dynamic Gaussian reconstruction for the entire scene while predicting which Gaussians should move.

6.2.2 *Timings*

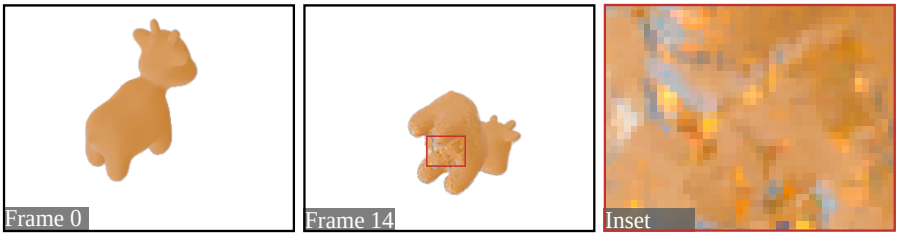
Although our method's inference is faster than competing methods, it is still not interactive due to the physics engine's speed. However, the modularity of our approach allows for the physics engine to be replaced by newer and faster alternatives, so advancements in physical simulation methods can be easily incorporated into our pipeline.

6.2.3 *Two-Stage Reconstruction Pipeline*

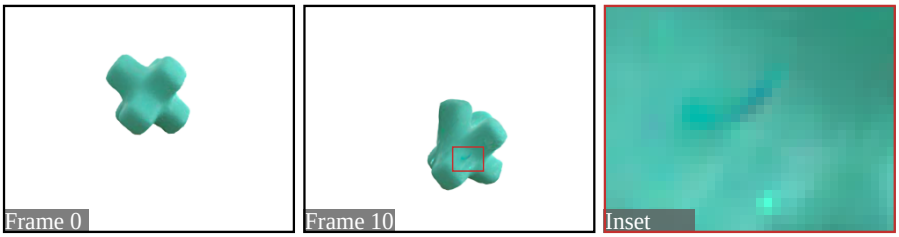
Our method relies on a two-stage pipeline for parameter recovery and reconstruction. This approach leads to the problems outlined in Section 6.2.6 since it is impossible to determine whether the reconstructed mesh has the required quality and resolution without incorporating the observed deformations in the mesh creation. However, note that this approach increases the robustness of our optimization, as optimizing both dynamics and topology simultaneously is difficult due to the many degrees of freedom involved. Furthermore, utilizing different stages makes our method modular, meaning that individual components can be replaced with better alternatives in the future. For instance, we have relied on traditional geometry processing techniques for mesh reconstruction, which could be replaced by recent alternatives targeted at mesh reconstruction from Gaussian splatting kernels [GL23].

6.2.4 *Static Appearance Reconstruction and Deformation Transfer*

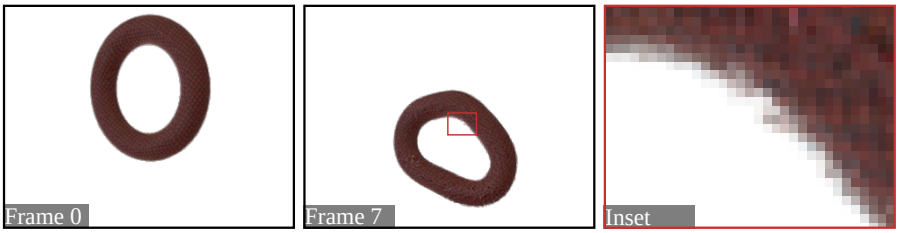
We observe three types of limitations in our appearance reconstruction. First, since our method optimizes appearance only for the first frame, areas of the object that become visible later are poorly reconstructed. Moreover, because of the deformation transfer, the relative order of the Gaussian kernels with regard to a viewpoint can change during the dynamic sequence, causing small colour changes to appear. Last, even though we rotate and scale the Gaussians based on the underlying deforming mesh, we see that sometimes small gaps between kernels can appear on the edges of objects. We demonstate the above types of artifacts in Figure 30.



(a) Unseen area artifacts.



(b) Relative order change artifacts.



(c) Kernel gap artifacts.

Figure 30: The different types of rendering artifacts produced by our method. We discuss potential solutions in this section.

All of the above limitations result from our method relying only on the first frame for appearance optimization. Therefore, a potential solution would be to use an extra pass over the multi-view video training sequence to fine-tune the deformation transfer. This fine tuning can include learning weights that associate each kernel to a tetrahedron, instead of relying solely on MVC, and further optimizing the kernels' colors based on the entire sequence.

6.2.5 *Ambiguity*

Although not directly an issue with our method, we observe that the training set significantly impacts the optimization landscape and, therefore, the physical parameter recovery. In Section 5.9, we demonstrated that the optimization landscape varies between scenes and provided supervision and specifically showed that for the datasets used, it does not exhibit a unique minimum, implying that the instances of physical parameter estimation problem we deal with are under-constrained. Since the analysis in Section 5.7 did not find any other significant source of error in our method, this ambiguity likely is the primary cause of errors in the physical parameter estimation.

The simplest solution to this is increasing the amount of training data used, while it is also interesting to investigate potential regularization approaches for the recovered parameter values. Furthermore, it is worth conducting more experiments to understand better how each observed deformation state affects the optimization landscape and attempt to quantify the uncertainty of the parameter values. The latter part has been recently done for the analogous problem in novel view synthesis [GRS*24], but to the best of our knowledge, it has yet to be explored in the inverse physics context. By quantifying the ambiguity, we reason that it will be possible to develop techniques to guide the data capturing process of real objects, making methods like ours more robust.

6.2.6 *Impact of Reconstructed Mesh*

As shown in Section 5.8, the quality of the reconstructed mesh may lead to dynamics different than those observed, even with the correct physical parameters. This issue is fundamental to using FEM-based simulations and, in principle, could be reduced by increasing the resolution of the mesh. Unfortunately, increasing the mesh's resolution imposes a significant increase in the GPU memory required to perform the inverse physics optimization. Last, as noted in Section 5.8, the mesh impacts the simulated dynamics but does not significantly affect the estimation of the physical parameter values.

One potential solution is adapting our technique to work with a meshless simulator, like the one used in PAC-NeRF. For this purpose, instead of reconstructing the mesh, particles can be sampled from the density field defined by the Gaussian kernels. In addition, this change would require adjusting the deformation transfer method so that the simulated particles can be used to guide the Gaussians.

A different solution to the limitations imposed by FEM is to investigate the inclusion of implicit approaches for learning dynamics that can predict corrective vectors for the mesh's vertices or directly dynamically remesh when needed (as done by Pfaff and colleagues in [PFSGB20]).

Last, we emphasize that our method allows for the mesh reconstruction step to be skipped if an alternative mesh can be provided. This can be beneficial in cases where a high quality 3D model of the object already exists or can be acquired through different means.

6.2.7 *Biases and Lack of Real Datasets*

In Section 5.4, we observed that our method and PAC-NeRF showcased better parameter estimation performance on data generated with their simulation. From this, we concluded that both techniques are biased in performing better on data that matches their physical

model. Experiments should be conducted with real data to address this and better understand our method's generalizability. Such an evaluation is currently challenging due to the lack of high-quality real datasets in this space. We reason these datasets do not exist because it is difficult to determine accurate elasticity parameters experimentally. However, we stress that effort should be put into creating such datasets since they offer the only way to understand the real-world generalizability of the physical models assumed by methods like ours.

6.3 CONCLUSIONS

We have presented a novel framework for joint appearance reconstruction and physical parameter estimation of elastic objects. Our method works on multi-view video input by utilizing dynamic 3D Gaussian splatting and differentiable Neo-Hookean simulation. We have demonstrated the ability of our method to estimate elasticity parameter values at the level of current state-of-the-art approaches while providing better visual quality and faster inference times. Our method represents a significant step forward in elastic object reconstruction, paving the way for accessible, realistic digital asset creation and we hope that it will inspire future research in this field.

BIBLIOGRAPHY

[BMT*21] Barron, Jonathan T, Mildenhall, Ben, Tancik, Matthew, Hedman, Peter, Martin-Brualla, Ricardo, and Srinivasan, Pratul P. "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields." *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, 5855–5864.

[BMV*22] Barron, Jonathan T, Mildenhall, Ben, Verbin, Dor, Srinivasan, Pratul P, and Hedman, Peter. "Mip-nerf 360: Unbounded anti-aliased neural radiance fields." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, 5470–5479.

[BKY*22a] Bergman, Alexander W., Kellnhofer, Petr, Yifan, Wang, Chan, Eric R., Lindell, David B., and Wetzstein, Gordon. "Generative Neural Articulated Radiance Fields." *NeurIPS*. 2022.

[BKY*22b] Bergman, Alexander, Kellnhofer, Petr, Yifan, Wang, Chan, Eric, Lindell, David, and Wetzstein, Gordon. "Generative neural articulated radiance fields." *Advances in Neural Information Processing Systems* 35 (2022), 19900–19916.

[BML*23] Bouaziz, Sofien, Martin, Sebastian, Liu, Tiantian, Kavan, Ladislav, and Pauly, Mark. "Projective dynamics: Fusing constraint projections for fast simulation." *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. 2023, 787–797.

[CTS*22] Chen, Hsiao-yu, Tretschk, Edith, Stuyck, Tuur, Kadlecek, Petr, Kavan, Ladislav, Vouga, Etienne, and Lassner, Christoph. "Virtual elastic objects." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, 15827–15837.

[CZB23] Chen, Huanyu, Zhao, Danyong, and Barbič, Jernej. "Capturing Animation-Ready Isotropic Materials Using Systematic Poking." *ACM Trans. Graph.* 42.6 (2023). issn: 0730-0301. doi: 10.1145/3618406. url: https://doi.org/10.1145/3618406.

[CCC*08] Cignoni, Paolo, Callieri, Marco, Corsini, Massimiliano, Dellepiane, Matteo, Ganovelli, Fabio, and Ranzuglia, Guido. "MeshLab: an Open-Source Mesh Processing Tool." *Eurographics Italian Chapter Conference*. Ed. by Scarano, Vittorio, Chiara, Rosario De, and Erra, Ugo. The Eurographics Association, 2008. isbn: 978-3-905673-68-5. doi: 10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136.

[Com18] Community, Blender Online. *Blender - a 3D modelling and rendering package*. Blender Foundation. Stichting Blender Foundation, Amsterdam, 2018. url: http://www.blender.org.

[CGH*20] Cranmer, Miles, Greydanus, Sam, Hoyer, Stephan, Battaglia, Peter, Spergel, David, and Ho, Shirley. "Lagrangian neural networks." *arXiv preprint arXiv:2003.04630* (2020).

[DHP24] DHPC, Delft High Performance Computing Centre. *DelftBlue Supercomputer (Phase 2)*. https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2. 2024.

[DWY*23] Das, Devikalyan, Wewer, Christopher, Yunus, Raza, Ilg, Eddy, and Lenssen, Jan Eric. "Neural parametric gaussians for monocular non-rigid object reconstruction." *arXiv preprint arXiv:2312.01196* (2023).

[DHD*19] Degrave, Jonas, Hermans, Michiel, Dambre, Joni, et al. "A differentiable physics engine for deep learning in robotics." *Frontiers in neurorobotics* 13 (2019), 406386.

[DWM*21] Du, Tao, Wu, Kui, Ma, Pingchuan, Wah, Sebastien, Spielberg, Andrew, Rus, Daniela, and Matusik, Wojciech. "Diffpd: Differentiable projective dynamics." *ACM Transactions on Graphics (TOG)* 41.2 (2021), 1–21.

[DMY*23] Duisterhof, Bardienus P, Mandi, Zhao, Yao, Yunchao, Liu, Jia-Wei, Shou, Mike Zheng, Song, Shuran, and Ichnowski, Jeffrey. "Md-splatting: Learning metric deformation from 4d gaussians in highly deformable scenes." *arXiv preprint arXiv:2312.00583* (2023).

[FYW*22] Fang, Jiemin, Yi, Taoran, Wang, Xinggang, Xie, Lingxi, Zhang, Xiaopeng, Liu, Wenyu, Niessner, Matthias, and Tian, Qi. "Fast dynamic radiance fields with time-aware neural voxels." *SIGGRAPH Asia 2022 Conference Papers*. 2022, 1–9.

[FCC*24] Feng, Qiyuan, Cao, Gengchen, Chen, Haoxiang, Mu, Tai-Jiang, Martin, Ralph R, and Hu, Shi-Min. "A New Split Algorithm for 3D Gaussian Splatting." *arXiv preprint arXiv:2403.09143* (2024).

[FSL*23] Feng, Yutao, Shang, Yintong, Li, Xuan, Shao, Tianjia, Jiang, Chenfanfu, and Yang, Yin. "PIE-NeRF: Physics-based Interactive Elastodynamics with NeRF." *arXiv preprint arXiv:2311.13099* (2023).

[FNPS16] Flynn, John, Neulander, Ivan, Philbin, James, and Snavely, Noah. "Deepstereo: Learning to predict new views from the world's imagery." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, 5515–5524.

[GSKH21] Gao, Chen, Saraf, Ayush, Kopf, Johannes, and Huang, Jia-Bin. "Dynamic view synthesis from dynamic monocular video." *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, 5712–5721.

[GYZ*24] Gao, Lin, Yang, Jie, Zhang, Bo-Tao, Sun, Jia-Mu, Yuan, Yu-Jie, Fu, Hongbo, and Lai, Yu-Kun. "Mesh-based Gaussian Splatting for Real-time Large-scale Deformation." *arXiv preprint arXiv:2402.04796* (2024).

[GRS*24] Goli, Lily, Reading, Cody, Sellán, Silvia, Jacobson, Alec, and Tagliasacchi, Andrea. "Bayes' Rays: Uncertainty Quantification in Neural Radiance Fields." *CVPR* (2024).

[GGSC23] Gortler, Steven J, Grzeszczuk, Radek, Szeliski, Richard, and Cohen, Michael F. "The lumigraph." *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. 2023, 453–464.

[GDY19] Greydanus, Samuel, Dzamba, Misko, and Yosinski, Jason. "Hamiltonian neural networks." *Advances in neural information processing systems* 32 (2019).

[GL23] Guédon, Antoine and Lepetit, Vincent. "Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering." *arXiv preprint arXiv:2311.12775* (2023).

[Han15] Hang, Si. "TetGen, a Delaunay-based quality tetrahedral mesh generator." *ACM Trans. Math. Softw* 41.2 (2015), 11.

[HPP*18] Hedman, Peter, Philip, Julien, Price, True, Frahm, Jan-Michael, Drettakis, George, and Brostow, Gabriel. "Deep blending for free-viewpoint image-based rendering." *ACM Transactions on Graphics (ToG)* 37.6 (2018), 1–15.

[HFG*18] Hu, Yuanming, Fang, Yu, Ge, Ziheng, Qu, Ziyin, Zhu, Yixin, Pradhana, Andre, and Jiang, Chenfanfu. "A moving least squares material point method with displacement discontinuity and two-way rigid body coupling." *ACM Transactions on Graphics (TOG)* 37.4 (2018), 1–14.

[HLS*19] Hu, Yuanming, Liu, Jiancheng, Spielberg, Andrew, Tenenbaum, Joshua B, Freeman, William T, Wu, Jiajun, Rus, Daniela, and Matusik, Wojciech. "Chainqueen: A real-time differentiable physical simulator for soft robotics." *2019 International conference on robotics and automation (ICRA)*. IEEE. 2019, 6265–6271.

[JKK*23] Jambon, Clément, Kerbl, Bernhard, Kopanas, Georgios, Diolatzis, Stavros, Leimkühler, Thomas, and Drettakis, George. "Nerfshop: Interactive editing of neural radiance fields." *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 6.1 (2023).

[JBH19] Jaques, Miguel, Burke, Michael, and Hospedales, Timothy. "Physics-as-inverse-graphics: Unsupervised physical parameter estimation from video." *arXiv preprint arXiv:1905.11169* (2019).

[JST*16] Jiang, Chenfanfu, Schroeder, Craig, Teran, Joseph, Stomakhin, Alexey, and Selle, Andrew. "The material point method for simulating continuum materials." *Acm siggraph 2016 courses*. 2016, 1–52.

[JYX*24] Jiang, Ying, Yu, Chang, Xie, Tianyi, Li, Xuan, Feng, Yutao, Wang, Huamin, Li, Minchen, Lau, Henry, Gao, Feng, Yang, Yin, et al. "VR-GS: A Physical Dynamics-Aware Interactive Gaussian Splatting System in Virtual Reality." *arXiv preprint arXiv:2401.16663* (2024).

[JSW23] Ju, Tao, Schaefer, Scott, and Warren, Joe. "Mean value coordinates for closed triangular meshes." *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. 2023, 223–228.

[KH13] Kazhdan, Michael and Hoppe, Hugues. "Screened poisson surface reconstruction." *ACM Transactions on Graphics (ToG)* 32.3 (2013), 1–13.

[KJJ*21] Kellnhofer, Petr, Jebe, Lars C, Jones, Andrew, Spicer, Ryan, Pulli, Kari, and Wetzstein, Gordon. "Neural lumigraph rendering." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, 4287–4297.

[KKLD23] Kerbl, Bernhard, Kopanas, Georgios, Leimkühler, Thomas, and Drettakis, George. "3d gaussian splatting for real-time radiance field rendering." *ACM Transactions on Graphics* 42.4 (2023), 1–14.

[KE20] Kim, Theodore and Eberle, David. "Dynamic deformables: implementation and production practicalities." *ACM SIGGRAPH 2020 Courses*. 2020, 1–182.

[KB15] Kingma, Diederik and Ba, Jimmy. "Adam: A Method for Stochastic Optimization." *International Conference on Learning Representations (ICLR)*. San Diega, CA, USA, 2015.

[KMR*23] Kirillov, Alexander, Mintun, Eric, Ravi, Nikhila, Mao, Hanzi, Rolland, Chloe, Gustafson, Laura, Xiao, Tete, Whitehead, Spencer, Berg, Alexander C, Lo, Wan-Yen, et al. "Segment anything." *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, 4015–4026.

[KLD23] Kratimenos, Agelos, Lei, Jiahui, and Daniilidis, Kostas. "Dynmf: Neural motion factorization for real-time dynamic view synthesis with 3d gaussian splatting." *arXiv preprint arXiv:2312.00112* (2023).

[LQC*23] Li, Xuan, Qiao, Yi-Ling, Chen, Peter Yichen, Jatavallabhula, Krishna Murthy, Lin, Ming, Jiang, Chenfanfu, and Gan, Chuang. "Pac-nerf: Physics augmented continuum neural radiance fields for geometry-agnostic system identification." *arXiv preprint arXiv:2303.05512* (2023).

[LWT*18] Li, Yunzhu, Wu, Jiajun, Tedrake, Russ, Tenenbaum, Joshua B, and Torralba, Antonio. "Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids." *arXiv preprint arXiv:1810.01566* (2018).

[LNSW21] Li, Zhengqi, Niklaus, Simon, Snavely, Noah, and Wang, Oliver. "Neural scene flow fields for space-time view synthesis of dynamic scenes." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, 6498–6508.

[LRS*21] Lin, Shanchuan, Ryabtsev, Andrey, Sengupta, Soumyadip, Curless, Brian L, Seitz, Steven M, and Kemelmacher-Shlizerman, Ira. "Real-time high-resolution background matting." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, 8762–8771.

[LMR*23] Loper, Matthew, Mahmood, Naureen, Romero, Javier, Pons-Moll, Gerard, and Black, Michael J. "SMPL: A skinned multi-person linear model." *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. 2023, 851–866.

[LJP*21] Lu, Lu, Jin, Pengzhan, Pang, Guofei, Zhang, Zhongqiang, and Karniadakis, George Em. "Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators." *Nature machine intelligence* 3.3 (2021), 218–229.

[LKLR24] Luiten, Jonathon, Kopanas, Georgios, Leibe, Bastian, and Ramanan, Deva. "Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis." *3DV*. 2024.

[MMC16] Macklin, Miles, Müller, Matthias, and Chentanez, Nuttapong. "XPBD: position-based simulation of compliant constrained dynamics." *Proceedings of the 9th International Conference on Motion in Games*. 2016, 49–54.

[MST*20] Mildenhall, B, Srinivasan, PP, Tancik, M, Barron, JT, Ramamoorthi, R, and Ng, R. "Nerf: Representing scenes as neural radiance fields for view synthesis." *European conference on computer vision*. 2020.

[MESK22] Müller, Thomas, Evans, Alex, Schied, Christoph, and Keller, Alexander. "Instant neural graphics primitives with a multiresolution hash encoding." *ACM transactions on graphics (TOG)* 41.4 (2022), 1–15.

[MS00] Murray-Smith, DJ. "The inverse simulation approach: a focused review of methods and applications." *Mathematics and computers in simulation* 53.4-6 (2000), 239–247.

[MMG*20] Murthy, J Krishna, Macklin, Miles, Golemo, Florian, Voleti, Vikram, Petrini, Linda, Weiss, Martin, Considine, Breandan, Parent-Lévesque, Jérôme, Xie, Kevin, Erleben, Kenny, et al. "gradsim: Differentiable simulation for system identification and visuomotor control." *International conference on learning representations*. 2020.

[NMK*06] Nealen, Andrew, Müller, Matthias, Keiser, Richard, Boxerman, Eddy, and Carlson, Mark. "Physically based deformable models in computer graphics." *Computer graphics forum*. Vol. 25. 4. Wiley Online Library. 2006, 809–836.

[PKK*24] Papantonakis, Panagiotis, Kopanas, Georgios, Kerbl, Bernhard, Lanvin, Alexandre, and Drettakis, George. "Reducing the Memory Footprint of 3D Gaussian Splatting." *Proceedings of the ACM on Computer Graphics and Interactive Techniques*. Vol. 7. 1. 2024.

[PSB*21] Park, Keunhong, Sinha, Utkarsh, Barron, Jonathan T, Bouaziz, Sofien, Goldman, Dan B, Seitz, Steven M, and Martin-Brualla, Ricardo. "Nerfies: Deformable neural radiance fields." *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, 5865–5874.

[PSH*21] Park, Keunhong, Sinha, Utkarsh, Hedman, Peter, Barron, Jonathan T, Bouaziz, Sofien, Goldman, Dan B, Martin-Brualla, Ricardo, and Seitz, Steven M. "HyperNeRF: a higher-dimensional representation for topologically varying neural radiance fields." *ACM Transactions on Graphics (TOG)* 40.6 (2021), 1–12.

[PGC*17] Paszke, Adam, Gross, Sam, Chintala, Soumith, Chanan, Gregory, Yang, Edward, DeVito, Zachary, Lin, Zeming, Desmaison, Alban, Antiga, Luca, and Lerer, Adam. "Automatic differentiation in PyTorch." (2017).

[PCG*19] Pavlakos, Georgios, Choutas, Vasileios, Ghorbani, Nima, Bolkart, Timo, Osman, Ahmed AA, Tzionas, Dimitrios, and Black, Michael J. "Expressive body capture: 3d hands, face, and body from a single image." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, 10975–10985.

[PYL*22] Peng, Yicong, Yan, Yichao, Liu, Shengqi, Cheng, Yuhao, Guan, Shanyan, Pan, Bowen, Zhai, Guangtao, and Yang, Xiaokang. "Cagenerf: Cage-based neural radiance field for generalized 3d deformation and animation." *Advances in Neural Information Processing Systems* 35 (2022), 31402–31415.

[PFSGB20] Pfaff, Tobias, Fortunato, Meire, Sanchez-Gonzalez, Alvaro, and Battaglia, Peter. "Learning Mesh-Based Simulation with Graph Networks." *International Conference on Learning Representations*. 2020.

[PAG*23] Pillonetto, Gianluigi, Aravkin, Aleksandr, Gedon, Daniel, Ljung, Lennart, Ribeiro, Antônio H, and Schön, Thomas B. "Deep networks for system identification: a survey." *arXiv preprint arXiv:2301.12832* (2023).

[PRLH*22] Portaneri, Cédric, Rouxel-Labbé, Mael, Hemmer, Michael, Cohen-Steiner, David, and Alliez, Pierre. "Alpha wrapping with an offset." *ACM Transactions on Graphics (TOG)* 41.4 (2022), 1–22.

[PCPMMN21] Pumarola, Albert, Corona, Enric, Pons-Moll, Gerard, and Moreno-Noguer, Francesc. "D-nerf: Neural radiance fields for dynamic scenes." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, 10318–10327.

[RSP*24] Radl, Lukas, Steiner, Michael, Parger, Mathias, Weinrauch, Alexander, Kerbl, Bernhard, and Steinberger, Markus. "StopThePop: Sorted Gaussian Splatting for View-Consistent Real-time Rendering." *arXiv preprint arXiv:2402.00525* (2024).

[SGGP*20] Sanchez-Gonzalez, Alvaro, Godwin, Jonathan, Pfaff, Tobias, Ying, Rex, Leskovec, Jure, and Battaglia, Peter. "Learning to simulate complex physics with graph networks." *International conference on machine learning*. PMLR. 2020, 8459–8468.

[SF16] Schonberger, Johannes L and Frahm, Jan-Michael. "Structure-from-motion revisited." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, 4104–4113.

[SL19] Schoukens, Johan and Ljung, Lennart. "Nonlinear system identification: A user-oriented road map." *IEEE Control Systems Magazine* 39.6 (2019), 28–99.

[SCD*06] Seitz, Steven M, Curless, Brian, Diebel, James, Scharstein, Daniel, and Szeliski, Richard. "A comparison and evaluation of multi-view stereo reconstruction algorithms." *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*. Vol. 1. IEEE. 2006, 519–528.

[SLK*20] Sengupta, Agniva, Lagneau, Romain, Krupa, Alexandre, Marchand, Eric, and Marchal, Maud. "Simultaneous tracking and elasticity parameter estimation of deformable objects." *2020 ieee international conference on robotics and automation (icra)*. IEEE. 2020, 10038–10044.

[SGK18] Smith, Breannan, Goes, Fernando De, and Kim, Theodore. "Stable neo-hookean flesh simulation." *ACM Transactions on Graphics (TOG)* 37.2 (2018), 1–15.

[SC23] Stuyck, Tuur and Chen, Hsiao-yu. "Diffxpbd: Differentiable position-based simulation of compliant constraint dynamics." *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 6.3 (2023), 1–14.

[SP04] Sumner, Robert W and Popović, Jovan. "Deformation transfer for triangle meshes." *ACM Transactions on graphics (TOG)* 23.3 (2004), 399–405.

[SSC22] Sun, Cheng, Sun, Min, and Chen, Hwann-Tzong. "Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, 5459–5469.

[TPBF87] Terzopoulos, Demetri, Platt, John, Barr, Alan, and Fleischer, Kurt. "Elastically deformable models." *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. 1987, 205–214.

[TFT*20] Tewari, Ayush, Fried, Ohad, Thies, Justus, Sitzmann, Vincent, Lombardi, Stephen, Sunkavalli, Kalyan, Martin-Brualla, Ricardo, Simon, Tomas, Saragih, Jason, Niessner, Matthias, et al. "State of the art on neural rendering." *Computer Graphics Forum*. Vol. 39. 2. Wiley Online Library. 2020, 701–727.

[TRJ*19] Toth, Peter, Rezende, Danilo Jimenez, Jaegle, Andrew, Racanière, Sébastien, Botev, Aleksandar, and Higgins, Irina. "Hamiltonian generative networks." *arXiv preprint arXiv:1909.13789* (2019).

[TTG*21] Tretschk, Edgar, Tewari, Ayush, Golyanik, Vladislav, Zollhöfer, Michael, Lassner, Christoph, and Theobalt, Christian. "Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video." *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, 12959–12970.

[TKB*23] Tretschk, Edith, Kairanda, Navami, BR, Mallikarjun, Dabral, Rishabh, Kortylewski, Adam, Egger, Bernhard, Habermann, Marc, Fua, Pascal, Theobalt, Christian, and Golyanik, Vladislav. "State of the Art in Dense Monocular Non-Rigid 3D Reconstruction." *Computer Graphics Forum*. Vol. 42. 2. Wiley Online Library. 2023, 485–520.

[UEK24] Uzolas, Lukas, Eisemann, Elmar, and Kellnhofer, Petr. "Template-free Articulated Neural Point Clouds for Reposable View Synthesis." *Advances in Neural Information Processing Systems* 36 (2024).

[WBT*24] Waczyńska, Joanna, Borycki, Piotr, Tadeja, Sławomir, Tabor, Jacek, and Spurek, Przemysław. "GaMeS: Mesh-Based Adapting and Modification of Gaussian Splatting." *arXiv preprint arXiv:2402.01459* (2024).

[WWY*15] Wang, Bin, Wu, Longhua, Yin, KangKang, Ascher, Uri M, Liu, Libin, and Huang, Hui. "Deformation capture and modeling of soft objects." *ACM Trans. Graph.* 34.4 (2015), 94–1.

[WELG21] Wang, Chaoyang, Eckart, Ben, Lucey, Simon, and Gallo, Orazio. "Neural trajectory fields for dynamic novel view synthesis." *arXiv preprint arXiv:2105.05994* (2021).

[WSND*23] Wang, Zian, Shen, Tianchang, Nimier-David, Merlin, Sharp, Nicholas, Gao, Jun, Keller, Alexander, Fidler, Sanja, Müller, Thomas, and Gojcic, Zan. "Adaptive shells for efficient neural radiance field rendering." *arXiv preprint arXiv:2311.10091* (2023).

[WJX*22] Willard, Jared, Jia, Xiaowei, Xu, Shaoming, Steinbach, Michael, and Kumar, Vipin. "Integrating scientific knowledge with machine learning for engineering and environmental systems." *ACM Computing Surveys* 55.4 (2022), 1–37.

[WYF*23] Wu, Guanjun, Yi, Taoran, Fang, Jiemin, Xie, Lingxi, Zhang, Xiaopeng, Wei, Wei, Liu, Wenyu, Tian, Qi, and Wang, Xinggang. "4d gaussian splatting for real-time dynamic scene rendering." *arXiv preprint arXiv:2310.08528* (2023).

[XHKK21] Xian, Wenqi, Huang, Jia-Bin, Kopf, Johannes, and Kim, Changil. "Space-time neural irradiance fields for free-viewpoint video." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, 9421–9431.

[XZQ*23] Xie, Tianyi, Zong, Zeshun, Qiu, Yuxin, Li, Xuan, Feng, Yutao, Yang, Yin, and Jiang, Chenfanfu. "Physgaussian: Physics-integrated 3d gaussians for generative dynamics." *arXiv preprint arXiv:2311.12198* (2023).

[XWZ*19] Xu, Zhenjia, Wu, Jiajun, Zeng, Andy, Tenenbaum, Joshua B, and Song, Shuran. "Densephysnet: Learning dense physical object representations via multi-step dynamic interactions." *arXiv preprint arXiv:1906.03853* (2019).

[YYPZ23] Yang, Zeyu, Yang, Hongye, Pan, Zijie, and Zhang, Li. "Real-time Photorealistic Dynamic Scene Representation and Rendering with 4D Gaussian Splatting." *The Twelfth International Conference on Learning Representations*. 2023.

[YHL*22] Yao, Chun-Han, Hung, Wei-Chih, Li, Yuanzhen, Rubinstein, Michael, Yang, Ming-Hsuan, and Jampani, Varun. "Lassie: Learning articulated shapes from sparse image ensemble via 3d part discovery." *Advances in Neural Information Processing Systems* 35 (2022), 15296–15308.

[YSL*22] Yuan, Yu-Jie, Sun, Yang-Tian, Lai, Yu-Kun, Ma, Yuewen, Jia, Rongfei, and Gao, Lin. "Nerf-editing: geometry editing of neural radiance fields." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, 18353–18364.

[ZIE*18] Zhang, Richard, Isola, Phillip, Efros, Alexei A, Shechtman, Eli, and Wang, Oliver. "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric." *CVPR*. 2018.

[ZYWL24] Zhong, Licheng, Yu, Hong-Xing, Wu, Jiajun, and Li, Yunzhu. "Reconstruction and Simulation of Elastic Objects with Spring-Mass 3D Gaussians." *arXiv preprint arXiv:2403.09434* (2024).

[ZTS*16] Zhou, Tinghui, Tulsiani, Shubham, Sun, Weilun, Malik, Jitendra, and Efros, Alexei A. "View synthesis by appearance flow." *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*. Springer. 2016, 286–301.

[ZBS*23] Zielonka, Wojciech, Bagautdinov, Timur, Saito, Shunsuke, Zollhöfer, Michael, Thies, Justus, and Romero, Javier. "Drivable 3d gaussian avatars." *arXiv preprint arXiv:2311.08581* (2023).

[ZPVBG02] Zwicker, Matthias, Pfister, Hanspeter, Van Baar, Jeroen, and Gross, Markus. "EWA splatting." *IEEE Transactions on Visualization and Computer Graphics* 8.3 (2002), 223–238.