Recovery of Herschel-Bulkley Fluid Parameters from Video via Differentiable Simulations

by

John M. Eastman

S.B. Computer Science and Engineering, Massachusetts Institute of Technology, 2023

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2024

Authored by: John M. Eastman
Department of Electrical Engineering and Computer Science
August 9, 2024

Certified by: Wojciech Matusik
Professor of Electrical Engineering and Computer Science, Thesis Supervisor

Accepted by: Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Recovery of Herschel-Bulkley Fluid Parameters from Video via Differentiable Simulations

by

John M. Eastman

Submitted to the Department of Electrical Engineering and Computer Science
on August 9, 2024 in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

ABSTRACT

Recreating the physical behavior of fluids from real-world footage remains a significant challenge, particularly for non-Newtonian fluids. This work introduces a novel method that combines neural radiance fields (NeRF), which map 3D scene coordinates to color and density using deep neural networks, with the material point method (MPM), a simulation technique that represents materials as moving points capable of large deformation. Our approach aims to accurately recover physical parameters and achieve high-fidelity 3D reconstructions from single-view videos of fluids, even those with complex rheological behaviors like shear thinning and thickening.

In this study, we apply our method to a Herschel-Bulkley fluid, namely ketchup, under two different real-world conditions: a 50mm column collapse and being squeezed from a bottle. By leveraging the differentiable nature of NeRF and the fluid simulation capabilities of MPM, our approach extracts parameters from real-world footage after initially training on approximate geometry derived from virtual models. The actual video footage is then used to estimate initial velocities and retrieve constitutive parameters, including modulus, yield stress, and viscosity. The iterative optimization process, which integrates continuous feedback between the NeRF-MPM simulation and the video data, enables us to extract constitutive parameters from real footage and perform predictive simulations that closely reflect the behavior observed in the training videos.

Key results include the retrieval of constitutive parameters, such as modulus, yield stress, and viscosity, as well as reconstructed videos that reflect the fluid behavior observed in the training video. The results demonstrate that our method can reconstruct the fluid's flow behavior from limited perspectives, accurately enough to visually reproduce the flow, showcasing its flexibility and robustness. This work not only validates the approach through

a series of experiments but also highlights the potential for differentiable rendering and simulation techniques to advance our understanding and simulation of complex material dynamics, particularly in cases where direct measurements are challenging or impossible.

Thesis supervisor: Wojciech Matusik
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

I would like to thank my Thesis Supervisor, Professor Wojciech Matusik, for his role in overseeing this work. I am especially grateful to my postdoc mentors, Peter Chen and Crystal Owens, for their invaluable guidance and support throughout this project.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recent advancements in differentiable rendering and simulation have significantly improved the ability to recover material properties and reconstruct 3D scenes from real-world video data.[1][2] These techniques are particularly promising for studying non-Newtonian fluids, which present complex rheological behaviors that are challenging to model using traditional methods. In contrast to Newtonian fluids, which have a constant viscosity, non-Newtonian fluids exhibit variable flow characteristics depending on the applied stress, leading to behaviors such as shear-thinning, shear-thickening, or yield stress. Accurately capturing these behaviors from visual data is crucial for applications ranging from industrial processes to biological systems.

This work builds upon the physics augmented continuum neural radiance fields (PAC-NeRF) framework,[2] which combines differentiable rendering with physical simulations to recover both the geometry and material properties of dynamic scenes. By adapting and extending PAC-NeRF, a novel pipeline has been developed to analyze real-world videos of non-Newtonian fluids, specifically focusing on the recovery of key constitutive parameters such as yield stress and plastic viscosity. This approach enables the accurate reconstruction of fluid dynamics from limited input data, such as single-view video footage, making it a powerful tool for studying complex material behaviors.

The motivation for this research stems from the need to better understand and predict the behavior of non-Newtonian fluids in real-world scenarios. The ability to extract material properties from visual data has broad implications, ranging from the optimization of manufacturing processes and the development of new materials to the analysis of biological systems. By leveraging differentiable simulation techniques, this work aims to provide a framework that can be generalized to a variety of non-Newtonian fluids, offering insights into their behavior under different conditions.

To validate the approach, a series of experiments were conducted using ketchup, a well-

known non-Newtonian fluid, in two distinct configurations: collapsing a 50mm column of ketchup and squeezing ketchup from a bottle. These experiments were chosen for their simplicity and their ability to demonstrate the characteristic behaviors of non-Newtonian fluids. The real-world video footage was processed using the adapted PAC-NeRF pipeline, enabling the extraction of material properties and the reconstruction of fluid dynamics. Additionally, forward simulations were performed to further explore the effects of varying these properties on the fluid's behavior, providing a deeper understanding of the underlying physics. These experiments not only validate the proposed pipeline but also provide insights that could be applied to a wide range of industrial and biological applications, showcasing the potential of this approach.

Chapter 2

Related Works

2.1 Differentiable Rendering

Differentiable rendering is a computational technique that allows the gradients of a rendering function to be computed with respect to its inputs. This capability is crucial for various applications in computer vision and graphics, as it enables the optimization of scene parameters directly from image data. By making the rendering process differentiable, it becomes possible to use gradient-based optimization techniques to refine models, adjust lighting, and recover material properties from observed images. In the context of 3D scene reconstruction, differentiable rendering offers a powerful framework for understanding how different scene elements, such as geometry, materials, and lighting, contribute to the final rendered image. This framework is particularly useful for tasks that require precise control and optimization, such as inverse rendering, where the goal is to deduce scene characteristics from images.

One form of differentiable rendering, Neural Radiance Fields (NeRFs), have become widespread in their use due to their robust functionality and high accuracy compared to previous methods. [3] Developed by Mildenhall et al., NeRF utilizes deep learning to create a volumetric representation of a scene, enabling the generation of photorealistic images from any viewpoint, provided there is a set of 2D images and corresponding camera poses.

NeRF uses a fully-connected deep neural network to represent a scene. The network takes a 5D coordinate as input, consisting of a spatial location and a viewing direction. It outputs the volume density and the view-dependent radiance for that location. The core of NeRF's approach involves querying these 5D coordinates along the rays cast from a virtual camera. By using volume rendering techniques, the outputs, colors and densities, are integrated along these rays to produce an image. Since the process of volume rendering is differentiable, it allows for the optimization of the scene representation using just a set of images with known camera poses.

NeRF has also been adapted to not only represent static scenes, but also dynamic scenes through methods such as Nerfies [4] and D-NeRF [5]. Nerfies extend the original NeRF framework to handle non-rigidly deforming objects, capturing both the scene's geometry and the temporal changes that occur within it. This adaptation allows for more complex and realistic reconstructions of scenes where the subjects are moving or changing over time, broadening the applicability of NeRF to a wider range of scenarios. Other approaches, like NR-NeRF, extend this to training a dynamic scene using monocular video recordings. [6]

NeRF is useful in generating photorealistic images from sparse input views. It excels in scenarios where 2D inputs are available but 3D output is desired, offering high-quality novel view synthesis. For our project, NeRF's ability to accurately capture and render scenes from limited input data is invaluable, especially in reconstructing scenes with complex fluid dynamics and materials.

2.2 Differentiable Simulation

Differentiable simulation is a computational approach that combines physical simulations with differentiable programming techniques. This allows for the calculation of gradients with respect to simulation parameters, enabling optimization and learning directly from observed data. Differentiable simulation is particularly valuable in scenarios where understanding and replicating complex physical phenomena is essential, as it provides a means to fine-tune simulations to closely match real-world observations. By leveraging differentiable programming, these simulations can optimize parameters like material properties, initial conditions, and external forces to achieve more accurate and realistic outcomes.

End-to-end differentiable physics engines, such as the one introduced by Belbute-Peres et al. [7], have made significant strides by facilitating the integration of physical simulations within neural networks, enhancing tasks like system identification, control, and prediction. The Affine Particle-In-Cell (APIC) method by Jiang et al. [8], an advancement of the material point method (MPM), is designed to make fluid simulations more stable and realistic. APIC reduces energy loss and preserves the natural motion of materials, integrating well with gradient-based optimization techniques used in modern simulations. Jiang et al. further expanded the application of MPM to simulating continuum materials, effectively handling large deformations and complex material interactions. [9].

In recent years, there has been a growing interest in integrating neural networks with PDE solvers to improve their accuracy. The "Solver-in-the-Loop" framework by Um et al. [10] allows neural networks to interact with PDE solvers during training, taking previous corrections into account and thus enhancing the accuracy of iterative solutions. Similarly, "Grad-

Sim" [11] presents a unified framework for differentiable physics and rendering, enabling the estimation of physical properties directly from video without requiring 3D supervision. Another notable contribution is "DiffPD" [12], a differentiable soft-body simulator based on Projective Dynamics (PD), which significantly speeds up simulations and is particularly effective for system identification, motion planning, and real-to-sim transfer. "Virtual Elastic Objects" by Chen et al. [13] focuses on capturing and reconstructing deformable objects using a differentiable, particle-based simulator to optimize material parameters, enabling realistic simulations and novel interactions with soft bodies. Additionally, the work by Wang et al.[14] presents a framework for learning customized elastic and damping models directly from motion data. "PlasticityNet"[15] introduces a neural network-based approach for simulating plastic solid materials, such as metal, sand, and snow, by learning a generalizable plastic energy.

In fluid simulation, "NeuroFluid" [16] integrates fluid physical properties into neural rendering using a particle-driven approach in the form of their proposed PhysNeRF, a particle-driven NeRF, advancing the unsupervised learning of fluid dynamics directly from visual observations. "Fluidlab" [17] presents their differentiable fluid physics engine "FluidEngine" paired with their Fluidlab environment designed to study fluid manipulation tasks in challenging settings, allowing one to study, optimize, and solve for these tasks.

Neural Constitutive Laws (NCLaw) [1] further integrates physics with differentiable simulation, combining neural networks with known PDE dynamics to ensure physical accuracy and generalizability. This approach is highly relevant to our work on PAC-NeRF, as it enhances the detailed 3D reconstruction and simulation of fluid dynamics.

Physics Augmented Continuum Neural Radiance Fields (PAC-NeRF)[2] represents an innovative extension of the NeRF framework and differentiable simulation MPM framework, designed to estimate both the unknown geometry and constitutive parameters of highly dynamic objects from multi-view videos. Introduced by Xuan Li et al., PAC-NeRF addresses the limitations of traditional NeRFs by utilizing the neural radiance field as a differentiable rendering tool to fit a learned simulation to physical parameters that match an input video sequence of a physical material, thereby creating new simulations that align closely with input video data.

PAC-NeRF employs a hybrid Eulerian-Lagrangian representation. In this setup, the Eulerian grid representation is used for the NeRF density and color fields, while the neural radiance fields are advected using Lagrangian particles. This hybrid approach seamlessly integrates efficient differentiable neural rendering with material point method (MPM) simulation for robust and differentiable physics simulation.

PAC-NeRF's key strength lies in its ability to model a wide range of dynamic scenarios

by capturing the physical properties and movements of objects. This method is particularly useful for simulating complex materials like fluids, which require precise modeling to achieve realistic results. My research focuses on extending PAC-NeRF to analyze real-world videos of viscous fluids, allowing for the accurate determination of material properties and detailed 3D reconstructions of fluid behavior.

2.3 Rheology of Non-Newtonian Fluids

The Herschel-Bulkley model [18] is an essential rheological framework for characterizing non-Newtonian fluids, which exhibit complex flow behaviors not adequately described by Newtonian or simple Bingham plastic models. This model accounts for a yield stress, τ_0, below which the material behaves as a solid and does not flow. Once the yield stress is exceeded, the fluid begins to flow, and the relationship between shear stress (τ) and shear rate ($\dot{\gamma}$) is described by a power-law equation:

$$\tau = \tau_0 + K\dot{\gamma}^n$$

where K is the consistency index and n is the flow index. This allows for the accurate modeling of both shear-thinning (pseudoplastic) and shear-thickening (dilatant) behaviors, depending on the value of n. When setting $n = 1$, the Bingham plastic model is recovered. The Herschel-Bulkley model is particularly useful for applications involving materials like pastes, gels, and slurries, which have significant yield stresses and non-linear flow properties. In the context of our research, this model provides a critical theoretical foundation for understanding and simulating the behavior of viscous fluids, aiding in the accurate recovery of material properties from real-world video data.

Advanced computational techniques, such as MPM, extend the applicability of the Herschel-Bulkley model to more complex fluid simulations. For example, the Continuum Foam framework by Yue et al. [19] uses MPM to model dense foams, which exhibit shear-dependent flows. This approach captures the viscoplastic and shear-thinning properties of foam, offering a robust tool for replicating intricate behaviors in materials like shaving cream and whipped cream.

Accurately modeling non-Newtonian fluids relies not only on robust theoretical frameworks but also on effective methods for estimating the material parameters that define these complex behaviors. The ability to capture and quantify these parameters from real-world observations is crucial for validating simulations and ensuring that the modeled behavior closely aligns with physical phenomena. Techniques like video-guided parameter transfer

and video-based rheometry provide innovative methods for extracting rheological properties from visual data. For example, Takahashi and Lin's framework [20] identifies material parameters of viscous fluids using video data, combining forward simulations with iterative parameter optimization. Similarly, the method presented in "Non-Newtonian ViRheometry via Similarity Analysis" [21] estimates Herschel-Bulkley parameters through video-based rheometry. Both methods require specific physical setups and controlled experimental conditions to achieve accurate estimations, which can limit their applicability in more generalized settings. However, they offer valuable insights into the rheology of non-Newtonian fluids and complement other methods used in studying complex fluid dynamics.

Chapter 3

Method

3.1 Theoretical Framework

In this section, we outline the theoretical foundations underlying our method. The approach leverages continuum mechanics principles, specifically focusing on the governing equations and constitutive models that describe the behavior of elastic, plastic, and fluid materials. We also detail the decomposition of the deformation gradient and the stress-strain relationships used in our simulations. This theoretical framework provides the basis for accurately modeling the physical properties and dynamic behavior of real-world materials within the PAC-NeRF framework.

3.1.1 Deformation Gradient and Tensor

The deformation gradient $F\mathbf{x}$ maps an infinitesimal vector in the undeformed configuration to the deformed configuration. For plasticity, F is decomposed into elastic F^e and plastic F^p parts

$$F = F^e F^p$$

3.1.2 Elasticity

Elasticity is described using the neo-Hookean model, characterized by the following stress-strain relationship. The Cauchy stress tensor $\mathbf{T}$ for neo-Hookean elasticity is defined by the equation:

$$J\mathbf{T}(\mathbf{F}) = \mu(\mathbf{F}\mathbf{F}^\top) + (\lambda \log(J) - \mu)\mathbf{I},$$

where μ and λ are the first and second Lame parameters, $\mathbf{F}$ is the deformation gradient, $J = \det(\mathbf{F})$ and $\mathbf{I}$ is the identity tensor. The Lame parameters are defined in relation to the Young's modulus E and Poisson's ratio ν by the equations

$$\mu = \frac{E}{2(1+\lambda)}, \lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}.$$

3.1.3 Plasticine Model

For modeling plasticine objects, we use the definitions used by Li et. al in PAC-NeRF [2]. A plasticine object is represented by the St.Venant-Kirchhoff (StVK) constitutive model using the von Mises yield condition for plastic return mapping. In this model, the Cauchy stress is defined by the following equation:

$$J\mathbf{T}(\mathbf{F}) = \mathbf{U}\exp(2\mu\boldsymbol{\epsilon} + \lambda \mathrm{Tr}(\boldsymbol{\epsilon}))\mathbf{U}^\top,$$

where $\mathbf{F}$ is the deformation gradient, $\mathbf{F} = \mathbf{U}\Sigma\mathbf{V}$ is the SVD decomposition of $\mathbf{F}$, and $\boldsymbol{\epsilon}$ is the Hencky strain, defined as $\boldsymbol{\epsilon} = \log(\Sigma)$.

The von Mises yielding condition is defined as

$$\delta\gamma = \|\hat{\boldsymbol{\epsilon}}\| - \frac{\tau_Y}{2\mu} > 0, \tag{3.1}$$

where $\hat{\boldsymbol{\epsilon}}$ is the normalized Hencky strain and τ_Y is the yield stress of the material.

3.1.4 Non-Newtonian Fluid Model

For modeling Non-Newtonian Fluid, we use the viscoplastic model defined by Yue et. al [19] in combination with the von Mises yield condition for defining the elastic vs. plastic regions. For this model, we use the following definitions from PAC-NeRF [2]

$$\hat{\mu} = \frac{\mu}{d}\text{Tr}(\Sigma^2)$$
$$\mathbf{s} = 2\mu\hat{\epsilon}$$
$$\hat{s} = -\|\mathbf{s}\| - \frac{\delta\gamma}{1 + \frac{\eta}{2\hat{\mu}\Delta t}}$$

with the return mapping defined as

$$\mathcal{Z}(\mathbf{F}) = \begin{cases} \mathbf{F} & \delta\gamma \leq 0 \\ \mathbf{U}\exp(\frac{\hat{s}}{2\mu}\hat{\boldsymbol{\epsilon}} + \frac{1}{d}\text{Tr}(\boldsymbol{\epsilon})\mathbf{1})\mathbf{V}^\top & \delta\gamma > 0 \end{cases} \tag{3.2}$$

3.2 Background: PAC-NeRF

PAC-NeRF, introduced by Xuan Li et al. [2] forms the basis for reconstruction and simulation. PAC-NeRF utilizes differentiable rendering, in the form of NeRF, combined with the MPM simulation method to recover the constitutive parameters, such as modulus, yield stress, and viscosity, from training videos.

3.2.1 Neural Radiance Fields

The differentiable rendering system that PAC-NeRF uses is based off of that from NeRF from Mildenhall et. al. [3] and Sun et. al. [22] NeRF presents a rendering system that is capable of being stored within the weights of a fully-connected deep neural network. A continuous scene is represented in NeRF as a NeRF takes as input a single 5D coordinate, composed of three spatial coordinates $\mathbf{x} = (x, y, z)$ and two viewing directions $\omega = (\theta, \phi)$, and outputs a radiance $\mathbf{c}$ and a volume density σ at the given location. NeRF uses a ray-based approach, where rays are cast from a camera viewpoint and points are sampled along the ray in order to retrieve the density and color information, contributing to the overall color of the ray. A ray is defined as $\mathbf{r}(s) = \mathbf{o} + s\omega$, where $\mathbf{o}$ is the origin of the ray, $\mathbf{d}$ is the direction, and s is the time along the ray with near and far bounds s_n and s_f. Using this, the color of a ray is defined using a volume rendering function by [3]

$$\mathbf{C}(\mathbf{r}) = \int_{s_n}^{s_f} T(s)\sigma(\mathbf{r}(s))\mathbf{c}(\mathbf{r}(s), \omega)ds, \text{ where } T(s) = \exp\left(-\int_{s_n}^{s_f} \sigma(\mathbf{r}(\bar{s}))d\bar{s}\right)$$

This equation is extended in PAC-NeRF to include a time parameter for the NeRF, so that

the NeRF may deform over time. Additionally, a background color term is included for rays in which the background color is visible, denoted by c_{bg}

$$\mathbf{C}(\mathbf{r},t) = \int_{s_n}^{s_f} T(s,t)\sigma(\mathbf{r}(s),t)\mathbf{c}(\mathbf{r}(s),\omega,t)ds + \mathbf{c}_{bg}T(s_f,t) \tag{3.3}$$
$$\text{where } T(s,t) = \exp\left(-\int_{s_n}^{s_f} \sigma(\mathbf{r}(\bar{s}),t))d\bar{s}\right)$$

With the time term included, the NeRF can now be trained to fit a dynamic continuous scene. The resulting output color is enforced to match the video data using a rendering loss term, defined as [2]

$$\mathcal{L}_{\text{render}} = \frac{1}{N}\sum_{i=0}^{N-1}\frac{1}{|\mathcal{R}|}\sum_{r\in\mathcal{R}}\|\mathbf{C}(\mathbf{r},\mathbf{t_i}) - \hat{\mathbf{C}}(\mathbf{r},\mathbf{t_i})\|^2 \tag{3.4}$$

where $\hat{\mathbf{C}}(\mathbf{r},\mathbf{t_i})$ is the ground truth color of the input.

3.2.2 Material Point Method

In addition to matching the color of the input video data, the density field of the NeRF is enforced to match the conservation laws of the velocity field of the physical system it is representing as simulated by MPM.

$$\frac{D\sigma}{Dt} = 0,\ \frac{D\mathbf{c}}{Dt} = 0,\ \rho\frac{D\phi}{Dt} = \frac{\partial\phi}{\partial t} + \mathbf{v}\cdot\nabla\phi \tag{3.5}$$

where $\frac{D}{Dt}$ represents the material derivative, and the third equation is the material derivative of a time-dependent field. In these equations, ϕ represents the density field of the physical system and $\mathbf{v}$ represents the velocity field. Additionally, $\mathbf{v}$ must obey momentum conservation for continuum materials, following the equation

$$\rho\frac{D\mathbf{v}}{Dt} = \nabla\cdot\mathbf{T} + \rho g \tag{3.6}$$

where $\mathbf{T}$ is the Cauchy stress tensor and g is the acceleration from gravity. These equations are combined to form the basis of MPM and NeRF as it is applied in PAC-NeRF.

3.2.3 Particle Grid System

While NeRF utilizes an Eulerian grid frame to render to an image, MPM uses a Lagrangian particle representation to advect the particles that make up our geometry. Therefore, PAC-NeRF uses a hybrid particle-grid system to effectively use NeRF in connection to MPM. PAC-NeRF supplies functions to convert between an Eulerian grid (G) representation of a field $\mathcal{F}^G$ and a Lagrangian particle (P) representation of a field $\mathcal{F}^P$. These fields can be converted between each other using the equations

$$\mathcal{F}_p^P \approx \sum_i w_{ip} \mathcal{F}_i^G, \; \mathcal{F}_i^G \approx \frac{\sum_p w_{ip} \mathcal{F}_p^P}{\sum w_{ip}} \tag{3.7}$$

Here, p denotes the particles, i refers to the index of the grid node, and w_{ip} signifies the weight of the trilinear shape function defined at node i and evaluated at the position of particle p.

In the context of PAC-NeRF, the initial representation is that of the Eulerian voxel field as learned by a static NeRF on the initial frame of video from each camera perspective. This field is converted to a Langrangian particle frame using the equation from 3.7 and advected according to MPM.

3.2.4 Parameter Estimation and Optimization Process

PAC-NeRF begins with an algorithm for geometry identification. PAC-NeRF does not need to be given geometry, and instead uses the multiple camera angles in the given data to reconstruct the geometry of the subject. The algorithm utilizes multiple steps in its method: data preprocessing, geometry seeding, and system identification.

First, before PAC-NeRF can be run, the data must be processed for input. PAC-NeRF takes in an input of image data organized by camera and frame number within a video sequence. The intrinsic and extrinsic parameters of the camera are known and provided directly to PAC-NeRF. These images have a video matting framework run on them from Lin et al. [23] This allows PAC-NeRF to learn only from the subject of each of the images by masking out the backgrounds.

The second step of PAC-NeRF is geometry seeding. This involves training a static NeRF using the masked first frames from each of the provided cameras. This NeRF is trained using a coarse-to-fine strategy. The loss functions employed to train the NeRF include the rendering loss function in 3.4.

where N is the number of frames and $\hat{\mathbf{C}}(\mathbf{r}, \mathbf{t_i})$ is the ground truth color, as well as a surface regularizer function to improve reconstruction quality, defined as

$$\mathcal{L}_{\text{surf}} = \sum_{p} \text{clamp}(\alpha_p, 10^{-4}, 10^{-1})(\frac{\Delta x}{2})^2 \tag{3.8}$$

The third and final step is to then perform parameter optimization using the extracted geometry. Using the first few frames of the video, the initial velocity of the particles contained with the geometry is estimated. The constitutive parameters for these initial frames is guessed. Then, using the following frames where the subject of the videos collides with the floor, the constitutive parameters are updated using gradient-based optimization on the NeRF.

Algorithm 1 Workflow for Fluid Parameter Estimation

Input: Sequence of training images with camera and time information.
Output: Optimized fluid parameters.

1: **Prepare the Data**
   1. Apply matting network to remove background from training images.
   2. Precompute rays and store them for querying in NeRF.
2: **Initialize Static NeRF with Training Images**
   1. Train static NeRF to capture the geometry of the fluid.
3: **Estimate Initial Velocity**
   1. Train NeRF on the first few frames of footage to estimate initial velocity.
4: **Optimize Fluid Parameters**
   1. See **Algorithm 2: Detailed Fluid Parameter Optimization**.

Algorithm 2 Fluid Parameter Optimization

Warm-Up Phase:
for a predefined number of iterations **do**
    Simulate fluid behavior using current parameters over the first few frames.
    Compute the loss function by comparing simulated frames with real-world frames.
    Calculate gradients of the loss with respect to fluid parameters.
    Update fluid parameters using gradient descent.
end for
Full Training Phase:
for a predefined number of iterations **do**
    Simulate fluid behavior using current parameters across the entire video sequence.
    Compute the loss function by comparing simulated frames with real-world frames.
    Calculate gradients of the loss with respect to fluid parameters.
    Update fluid parameters using gradient descent.
end for

3.3 Single-view reconstruction

In this work, we focused on extracting constitutive parameters and reconstructing real-world video footage captured from a single view, specifically using an iPhone camera. The process began with determining the camera intrinsics properties using the checkerboard method, a common technique for camera calibration, implemented via OpenCV. Camera intrinsics refer to the internal characteristics of the camera, including the focal length, optical center (principal point), and lens distortion coefficients. These parameters are crucial for understanding how the camera interprets the 3D world onto a 2D image plane.

To estimate the camera extrinsic properties, which define the camera's position and orientation in space relative to the world coordinates, I utilized an approach based on image overlaying. Camera extrinsics consist of a rotation matrix and a translation vector, which together describe the transformation from the world coordinate system to the camera coordinate system. By comparing and tweaking the extrinsic matrices, I aligned the simulated footage with the real-world images, ensuring an accurate match between the virtual and actual scenes.

Training PAC-NeRF on single-view data required the approximate 3D geometry of the real-world subject. This was achieved by recreating the subject's geometry in 3D modeling software, carefully matching it to the real-world footage. Once the approximate geometry was established, PAC-NeRF was used to train the static NeRF component on the 3D simulation geometry. Following this, the velocity and dynamic aspects of the NeRF were trained using the actual real-world footage.

This method allowed for effective reconstruction and parameter extraction from single-view video data, leveraging the strengths of PAC-NeRF in both static and dynamic scene understanding. The assumption underlying this approach is that the approximate geometry created in the 3D modeling software is sufficiently accurate to enable PAC-NeRF to generalize from the simulated training data to the real-world dynamic footage.

3.4 Real-world Validation

For real-world validation of the algorithm, videos were captured of ketchup, a non-Newtonian fluid, in two distinct configurations to observe its behavior under different conditions. These configurations include experiments using a 3 oz paper cup and squeezing ketchup out of a bottle. The 3 oz paper cup was chosen to mimic a concrete slump test, a method proven valuable for predicting yield stress by observing the height of the resulting slump from a controlled column collapse, a visible characteristic of the experiment.[24] Footage for both setups was captured from a single perspective using an iPhone camera. The camera intrinsics were calibrated and are common to both experiments, ensuring consistent data capture. The camera extrinsics for each experiment were carefully aligned manually by comparing rendered images to the captured footage and using image overlays to ensure precise alignment.

3.4.1 Experimental Setup 1: 50mm Column Collapse

In the first experimental setup, a 3 oz paper cup was used to study the flow of ketchup. The dimensions of the cup were measured with calipers: the bottom diameter was 38mm, the top diameter was 54mm, and the height was 53mm. The inside of the cup was covered with a thin layer of vegetable oil, applied with a Q-Tip, to minimize adhesion. The cup was filled with ketchup to the brim with the excess skimmed off and then flipped upside down. After flipping, holes were cut in the bottom of the cup to allow air to flow into the cup and displace the ketchup as it flows out of the cup. The cup was then slowly raised, allowing the ketchup to gradually lose adhesion to the cup and eventually fall, simulating a controlled flow.

The images showing the initial setup, the middle of the process, and the end result can be seen in figure 3.1

3.4.2 Experimental Setup 2: Squeezing Ketchup from a Bottle

In the second experimental setup, the behavior of ketchup being squeezed from a bottle was captured. The ketchup bottle was positioned approximately 2.5 inches, or 6.35 centimeters,

Figure 3.1: Ketchup flow from a 3 oz paper cup: initial setup (left), middle of the process (center), and end result (right).

above the surface being squeezed onto. Video footage was recorded to capture the formation of ketchup streams, which collected into a single pile on the surface due to the high viscosity of the ketchup.

The images showing the initial setup, the middle of the process, and the end result can be seen in figure 3.2

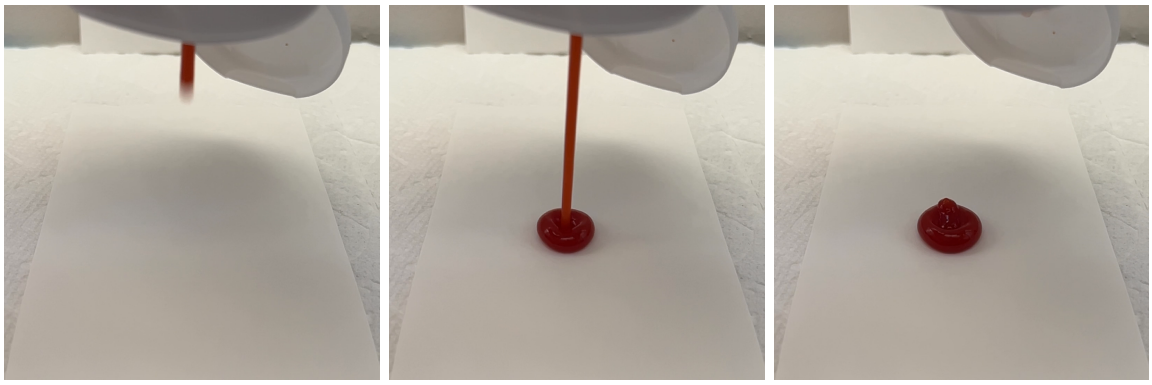


Figure 3.2: Squeezing ketchup from a bottle: initial setup (left), middle of the process (center), and end result (right).

Chapter 4

Results

4.1 Constitutive Model Validation

To validate the accuracy of the constitutive model, forward simulations were run using the Material Point Method (MPM) with various configurations of Herschel-Bulkley fluids. This section details the experiments conducted with simulated scenarios of ketchup, examining how well the constitutive model captures the behavior of the real-world material. Validating the constitutive model through these virtual experiments is crucial for demonstrating that the model can accurately represent the complex rheological behaviors of non-Newtonian fluids, ensuring its reliability when applied to real-world scenarios.

The simulation setup involved a cylinder of ketchup with a radius of 0.2 units and a height of 0.4 units. Various values of yield stress σ_y and plastic viscosity η were used to simulate the behavior of the ketchup. The simulations were run to observe the collapse of the cylinder and to measure the height of the resulting pile at a specific time, as well as the change in height over time.

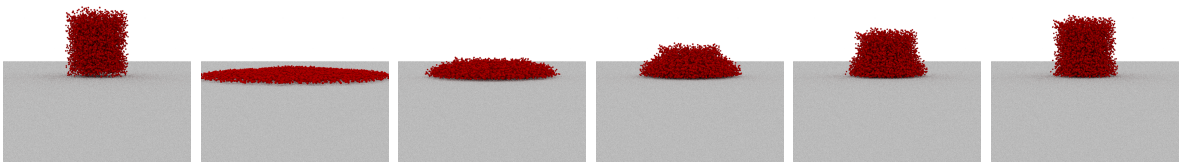
4.1.1 Varying Yield Stress

The height of the ketchup column at t=1s was measured for different values of yield stress σ_y, while keeping the plastic viscosity η constant with $\eta = 10$. The results are visually depicted in figure 4.1, with the values plotted in figure 4.2 and listed in table 4.1.

4.1.2 Varying Plastic Viscosity

The height of the ketchup column at t=1s was also measured for different values of plastic viscosity η, while keeping the yield stress σ_y constant with $\sigma_y = 1000$. The results are

visually depicted in figure 4.3, with the values plotted in figure 4.4 and listed in table 4.2.



(a) Initial (b) $\sigma_y = 100$ (c) $\sigma_y = 500$ (d) $\sigma_y = 1000$ (e) $\sigma_y = 2000$ (f) $\sigma_y = 5000$

Figure 4.1: Column collapse simulation at t=1s for various yield stress (σ_y) values.

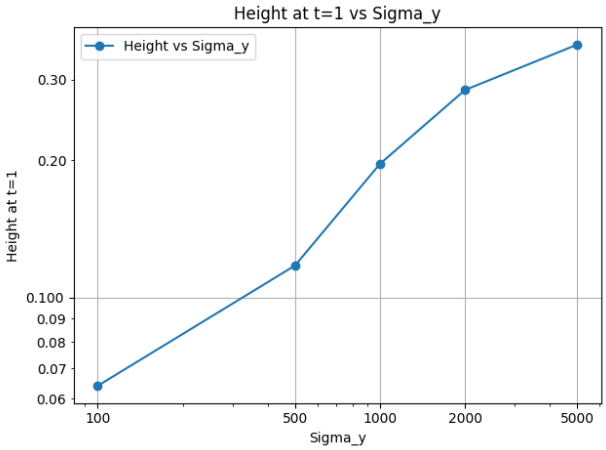
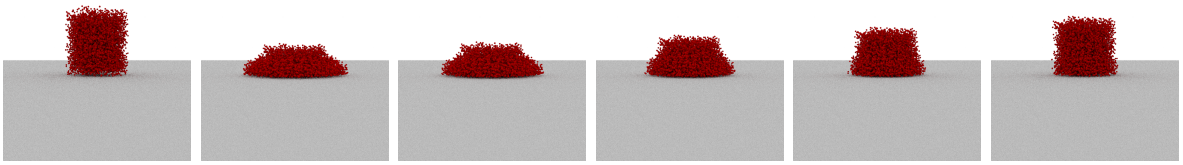


Figure 4.2: Plot of Height at t=1 vs σ_y

σ_y	Height at t=1s
100	0.064069
500	0.11758
1000	0.19640
2000	0.28506
5000	0.35848

Table 4.1: Height of the ketchup column at t=1s for different yield stress σ_y values.



(a) Initial (b) $\eta = 1$ (c) $\eta = 10$ (d) $\eta = 100$ (e) $\eta = 1000$ (f) $\eta = 10000$

Figure 4.3: Column collapse simulation at t=1s for various plastic viscosity (η) values.

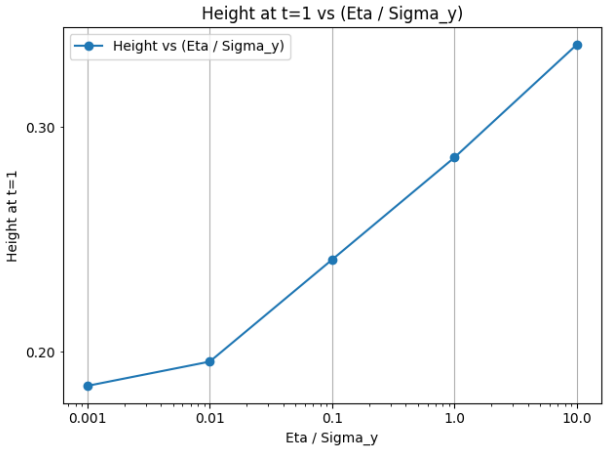


Figure 4.4: Plot of Height at t=1 vs η/σ_y

σ_y/η	Height at t=1s
0.001	0.18800
0.01	0.19640
0.1	0.23609
1.0	0.28390
10.0	0.34804

Table 4.2: Height of the ketchup column at t=1s for different plastic viscosity η values.

4.2 Virtual Experiments

In this section, we present the results of virtual experiments designed to mimic real-world experiments involving ketchup flow dynamics. The primary goal of these experiments is to validate the accuracy of the reconstruction algorithm in replicating the behavior observed in virtual physical scenarios. Specifically, we explore two setups that mirror our real-world experiments: the flow of ketchup from a 3 oz paper cup and the squirt of ketchup from a bottle. By adjusting simulation parameters such as yield stress, viscosity, and gravity, we aim to closely replicate the real-world behavior and extract accurate material properties from the simulated data.

4.2.1 Virtual Cup Experiment

The first virtual experiment mirrored the setup of the ketchup 50mm column collapse from the 3 oz paper cup. The simulation began with a virtual truncated cone of viscous fluid representing the ketchup, using scaled up dimensions of the real cup used for the real world experiment. The truncated cone is already flipped, allowing the fluid to flow from the start under the influence of gravity in the simulation. The simulation parameters, such as yield stress and viscosity, were adjusted to mimic the real-world behavior observed in the corresponding physical experiment. This simulation was run with the values $\mu = 1000$, $\lambda = 1000$, $\sigma_y = 50$, and $\eta = 1$. The figure 4.5 shows how the forward simulation progressed over time. This simulation was run for 0.5 seconds, sufficient for the cup to deform under its weight and settle.

Figure 4.5: Virtual simulation of ketchup flow from a 3 oz paper cup over time from beginning (left) to middle (center) and end (right).

The algorithm was then run on the video generated by the forward MPM simulation, using 11 camera angles for the full 12 frame sequence. The initialization of the simulation

with guesses for the constitutive parameters can be seen in figure 4.6, and the recovered results can be seen in figure 4.7. The extracted parameter values for the ground truth, guessed initial values, and final learned result can be seen in 4.3. These results show that parameter extraction is successful for the μ, σ_y, and η results, with the recovered value for λ being 74.8% off of the ground truth value.

Figure 4.6: Our initialization of a simulation of ketchup flow from a 3 oz paper cup over time from beginning (left) to middle (center) and end (right).

Figure 4.7: Our learned result of ketchup flow from a 3 oz paper cup over time from beginning (left) to middle (center) and end (right).

Parameter	Ground Truth	Initialization	Recovered	Percent Error
μ	1000	100	967	3.3%
λ	1000	10000	252	74.8%
σ_y	50	10	39	22%
η	1	1	1	0%

Table 4.3: Values of the constitutive parameters for the virtual cup experiment.

4.2.2 Virtual Squirt Experiment

The second virtual experiment simulated the behavior of ketchup being squirted from a bottle. In this virtual experiment, a prolate spheroid of viscous fluid was used to emulate the shape of a squirt of ketchup from a bottle. For this experiment, the squirt was scaled up in size to a spheroid with height 0.6 and radius 0.1 units. This object was positioned 0.5 units above the ground in the simulation, allowing it to accumulate velocity from gravity and impact the ground to form a pile of ketchup in the forward simulation. The simulation parameters were tuned to replicate the observed viscosity and flow characteristics of the ketchup during the experiment. The figure 4.8 shows how the forward simulation progressed over time. The simulation was run for 0.5 seconds.

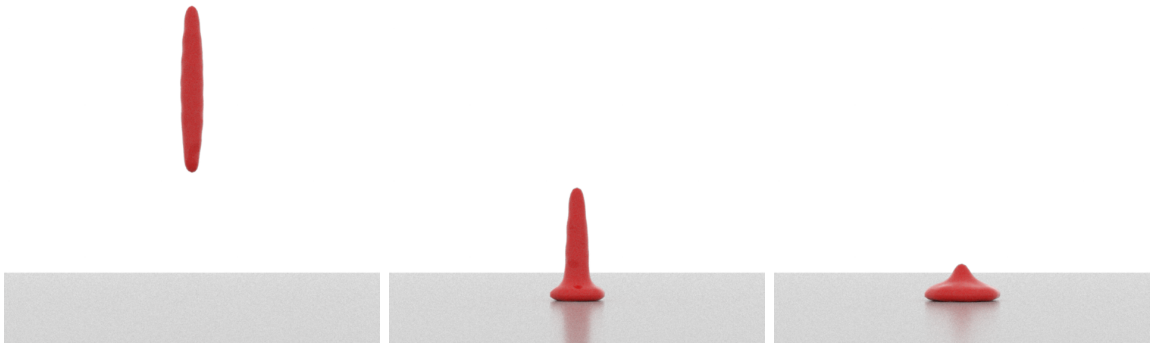


Figure 4.8: Virtual simulation of ketchup flow from a squirt over time from beginning (left) to middle (center) and end (right).

The algorithm was again run on the video generated by the forward MPM simulation, using 11 camera angles for the full 12 frame sequence. Our initialization of the simulation with guesses for the constitutive parameters for can be seen in figure 4.9, and the learned result can be seen in figure 4.10. The extracted parameter values for the ground truth, guessed initial values, and final learned result can be seen in table 4.4.

Parameter	Ground Truth	Initialization	Recovered	Percent Error
μ	100000	100	1971	98%
λ	100000	1000	32258	67.7%
σ_y	1500	100	372	75%
η	15	1	42	180%

Table 4.4: Values of the constitutive parameters for the virtual squeeze experiment.

We then run the same experiment, scaled down significantly in order to replicate a real-world ketchup squirt. In this configuration, the radius of the oblate spheroid is 0.004 units and the height of the squirt is 0.045. The forward simulation can be seen in figure 4.11.

Figure 4.9: Our initialization of a simulation of a ketchup squirt over time from beginning (left) to middle (center) and end (right).

Figure 4.10: Our learned result of a ketchup squirt over time from beginning (left) to middle (center) and end (right).

Figure 4.11: Virtual simulation of ketchup flow from a small squirt over time from beginning (left) to middle (center) and end (right).

The initialization of the simulation can be seen in figure 4.12 and the learned results can be seen in the figure 4.13. The ground truth, guessed initial values, and extracted parameters can be seen in table 4.5.

Figure 4.12: Our initialization of a simulation of a small ketchup squirt over time from beginning (left) to middle (center) and end (right).

Figure 4.13: Our learned result of a small ketchup squirt over time from beginning (left) to middle (center) and end (right).

Parameter	Ground Truth	Initialization	Recovered	Percent Error
μ	1000	100	786	21.4%
λ	1000	1000	1041	4.1%
σ_y	50	100	75	50%
η	1	10	0.2	80%

Table 4.5: Values of the constitutive parameters for the small virtual squeeze experiment.

4.3 Ketchup Cup Experiment

In this real-world experiment, ketchup was allowed to flow from a 3 oz paper cup, replicating the conditions detailed in the method section. The setup aimed to observe and measure the flow dynamics and final resting shape of the ketchup as it exited the cup and settled. The

footage of the real-world ketchup cup was captured for 2/3 of a second, allowing enough time for the cups initial deformation under its own weight to be recorded. The simulation was initialized with guesses for the constitutive parameters. The initialization of the algorithm can be seen in figure 4.15, and the final learned reconstruction can be seen in figure 4.16. The constitutive parameter values for the initial guess and the learned reconstruction are detailed in table 4.6. The ground truth values are not known for the real-world experiments, however, the values can still be extracted from the footage.

Figure 4.14: Real-world footage of ketchup cup over time from beginning (left) to middle (center) and end (right).

Figure 4.15: Our initialization of a simulation of real-world ketchup cup over time from beginning (left) to middle (center), and end (right).

Figure 4.16: Our learned result of real-world ketchup cup over time from beginning (left) to middle (center) and end (right).

Parameter	Initialization	Recovered
μ	1	564
λ	1	69670
σ_y	1	175
η	1	3.65

Table 4.6: Values of the constitutive parameters for the real-world cup experiment.

4.4 Ketchup Squeeze Experiment

In this real-world experiment, ketchup was squeezed from a bottle, replicating the conditions detailed in the method section. The setup aimed to observe the squirt of ketchup as it was squeezed from a bottle and landed on the ground surface. The footage of the real-world ketchup squeeze was captured for 2/3 of a second. The initialization of the algorithm can be seen in 4.18, and the final learned reconstruction can be seen in 4.19. The constitutive parameter values for the initial guess and the learned reconstruction are detailed in 4.7. The ground truth values are not known for the real-world experiments, however, the values can still be extracted from the footage.

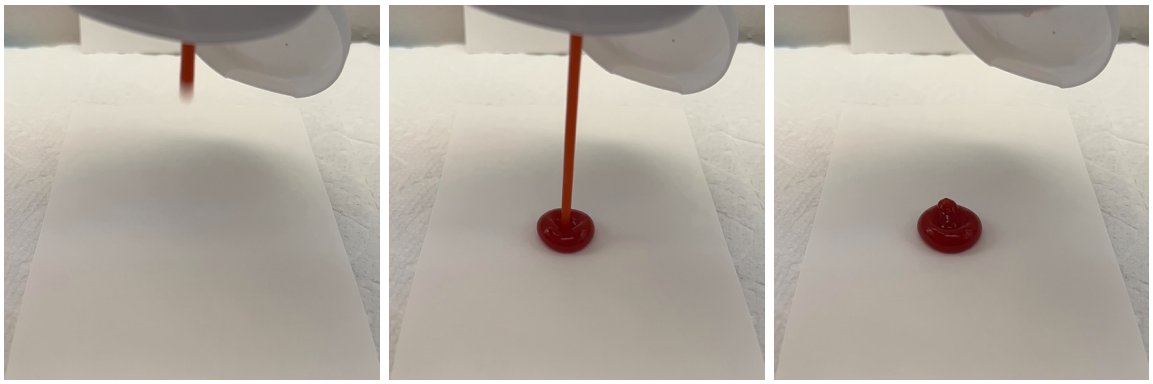


Figure 4.17: Real-world footage of ketchup squeeze over time from beginning (left) to middle (center) and end (right).

Parameter	Initialization	Recovered
μ	5000	885
λ	5000	5676
σ_y	200	28.6
η	10	0.32

Table 4.7: Values of the constitutive parameters for the real-world squeeze experiment.

Figure 4.18: Our initialization of a simulation of real-world ketchup squeeze over time from beginning (left) to middle (center) and end (right).

Figure 4.19: Our learned result of real-world ketchup squeeze over time from beginning (left) to middle (center) and end (right).

4.5 Novel Simulations: Generalization Beyond Training Data

Using the parameters recovered from reconstruction experiments, we can also simulate new scenes and situations with different geometry. Typical machine learning models do not perform well in generalization, due to fitting over the training data and scene specifically. Our model, however, due to the baked-in physics model and learning of material parameters, is able to generalize very well across different geometries, simulation times, boundary conditions, and multi-physics. To demonstrate this, we applied the material parameters obtained from the real-world ketchup simulation to a forward simulation involving a Stanford Bunny model.[25] This approach allows us to observe how the recovered non-Newtonian fluid properties behave when applied to a completely different geometry.

The simulation was carried out by initializing the Stanford Bunny with the recovered ketchup parameters from the ketchup squeeze in figure 4.7, then allowing the model to evolve under the influence of gravity and external forces. Figure 4.20 showcases the bunny's deformation over time, with images representing the initial state, the middle of the simulation, and the final state after the fluid properties have fully influenced the geometry.

Figure 4.20: Stanford Bunny simulation using recovered ketchup parameters over time from beginning (left) to middle (center) and end (right).

These results highlight the flexibility and capability of the recovered parameters, demonstrating their applicability not only to the original experiment but also to new and complex geometries. This opens up new possibilities for simulating non-Newtonian fluid behaviors across a wide range of contexts.

Chapter 5

Limitations and Future Work

5.1 Performance Evaluation

5.1.1 Strengths

One of the key strengths of the proposed framework is its ability to generate simulations where the physical behavior of the fluid appears very similar to the input video when observed by the naked eye. This visual fidelity in terms of fluid dynamics is crucial for applications where accurately portraying fluid behavior is essential, such as in animation or visual effects. The algorithm allows for high-quality reconstructions that effectively capture the dynamics of complex, non-Newtonian fluids like ketchup. The framework excels in producing simulations that, while not visually identical, closely mirror the real-world fluid behavior in a convincing manner.

The framework also performs particularly well in scenarios involving multiple view reconstruction, such as in virtual experiments. With multiple camera angles, the system can fully utilize the data available to create detailed and accurate simulations, showcasing its ability to handle complex fluid dynamics effectively.

5.1.2 Weaknesses

Despite its strengths in simulating fluid behavior, the framework can face challenges in accurately recovering certain physical parameters in some scenarios, specifically the lambda (λ) and plastic viscosity (η) parameters. These parameters are critical for characterizing the elastic and viscous properties of non-Newtonian fluids, and inaccuracies in their recovery can lead to deviations between the simulated and real-world material behaviors. The difficulty in accurately determining these parameters suggests that while the framework is effective for

producing visually convincing simulations, it may not always capture the underlying physical properties with the same level of precision.

Additionally, when using single-view real-world footage for reconstruction, the algorithm often required extensive tuning and adjustments. The single viewpoint likely constrained the framework's ability to fully capture and understand the complex geometry and fluid interactions present in the scene. This limitation in perspective reduces the available information, making it more difficult to accurately reconstruct the 3D structure and behavior of the fluid, leading to less accurate results. Furthermore, the framework also demands more parameter tuning and extended training times when working at very small scales, such as the ketchup squirt experiment, due to the need to reduce the grid size. These factors can increase the complexity and time required for the simulation process, particularly in applications involving fine details and small-scale fluid dynamics.

5.2 Future Work

5.2.1 Training a Unified Model from Multiple Data Trajectories

Another promising direction for future research is the development of a unified model that can be trained on multiple data trajectories simultaneously. For example, combining data from experiments involving both a cup and a squirting fluid could enable the model to generalize across different types of material behavior, such as varying shear rates and shear stresses. This approach would leverage the diverse range of dynamic behaviors captured in these trajectories to create a more versatile and robust model. By incorporating the relationships between shear rate and shear stress, the model could better predict material properties and behaviors across a broader spectrum of non-Newtonian fluids. This could lead to more accurate simulations and predictions in complex, real-world scenarios where multiple types of fluid dynamics are at play.

5.2.2 3D Gaussian Splatting as an Alternative to NeRF

While NeRF has proven effective in capturing and rendering complex scenes, recent advancements suggest that 3D Gaussian Splatting [26], another differentiable rendering algorithm, could offer a more efficient alternative. Unlike NeRF, which relies on dense sampling of the scene and complex volumetric rendering, 3D Gaussian Splatting uses a collection of Gaussian primitives that can be directly optimized to represent the scene's geometry and appearance. This approach has the potential to significantly reduce the computational burden, enabling

faster convergence and real-time applications. Future work could involve replacing the NeRF backbone with a 3D Gaussian Splatting framework, which may allow for more efficient rendering and potentially open up new possibilities for handling larger and more complex scenes with improved performance.

Chapter 6

Conclusion

This thesis presents a novel approach to recovering material properties of non-Newtonian fluids from real-world video data by extending the PAC-NeRF framework. Through a combination of differentiable rendering and physical simulation, the developed pipeline successfully extracts key constitutive parameters, such as yield stress and plastic viscosity, and reconstructs the fluid dynamics from limited input data. The approach was validated through a series of real-world and virtual experiments using ketchup as a representative non-Newtonian fluid. The results demonstrate the method's ability to accurately capture the complex rheological behaviors characteristic of non-Newtonian fluids.

The experimental findings show that the adapted PAC-NeRF framework is capable of not only reconstructing the fluid dynamics from single-view video footage but also of providing meaningful insights into the material properties that govern these dynamics. The virtual experiments closely mimicked the real-world scenarios, reinforcing the algorithm's effectiveness and generalizability. However, the results also highlighted the challenges in accurately recovering certain parameters, such as the parameter λ, suggesting areas for further refinement in the optimization process.

The implications of this work are significant for various fields, including industrial processing, materials science, and biological systems analysis. The ability to extract material properties from visual data opens up new possibilities for optimizing manufacturing processes, designing new materials, and studying complex fluid behaviors in natural and artificial systems. Future work could focus on improving the accuracy of parameter recovery, extending the method to handle a broader range of fluid types, and exploring its application to other dynamic scenes beyond non-Newtonian fluids.

In conclusion, this thesis contributes a robust and versatile tool for the study of non-Newtonian fluids, demonstrating the power of combining differentiable rendering with physical simulation. The methods and findings presented here pave the way for future research in

this area, offering a new perspective on the analysis and understanding of complex material behaviors from visual data.

Appendix A

Reconstruction Training Graphs

Included below are training graphs for each of the results. The loss curve represents the loss at each iteration of training, as defined by the equations in chapter 3 The change in loss at iteration 50 is due to the end of the warm-up phase, intended to reduce the chance of convergence issues. The learning rate curve shows the learning rate for each of the constitutive parameters during training, following the cosine annealing learning rate with a restart between the warm-up phase and the full training phase.[27] The four plots below the loss and learning rate plots show the parameters as they are optimized throughout the learning sequence. These parameter plots are, in order from top left to bottom right, μ, λ, σ_y, and η.

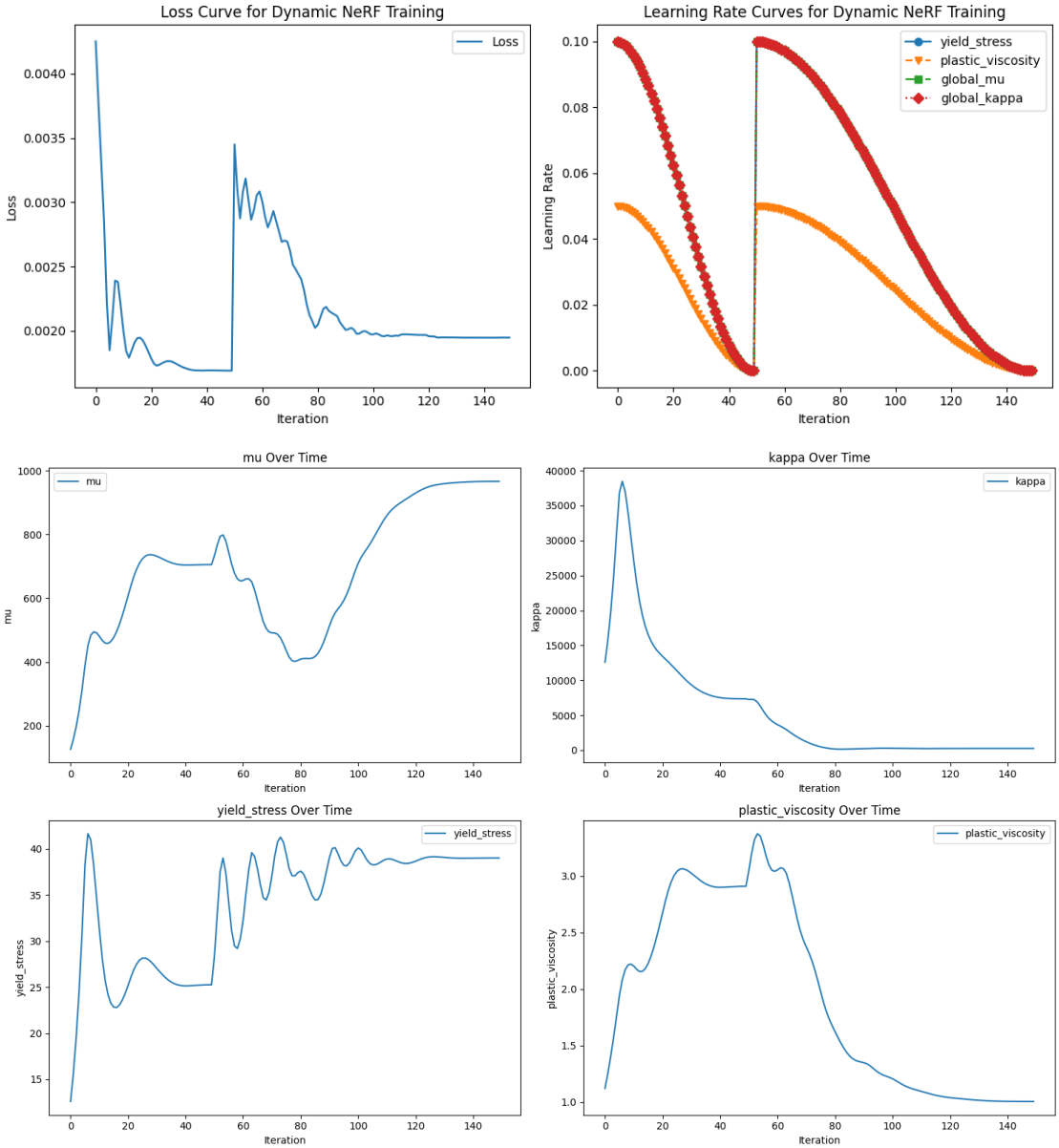


Figure A.1: Training plots for virtual ketchup cup: loss and learning rates (top two), material parameters (bottom four).

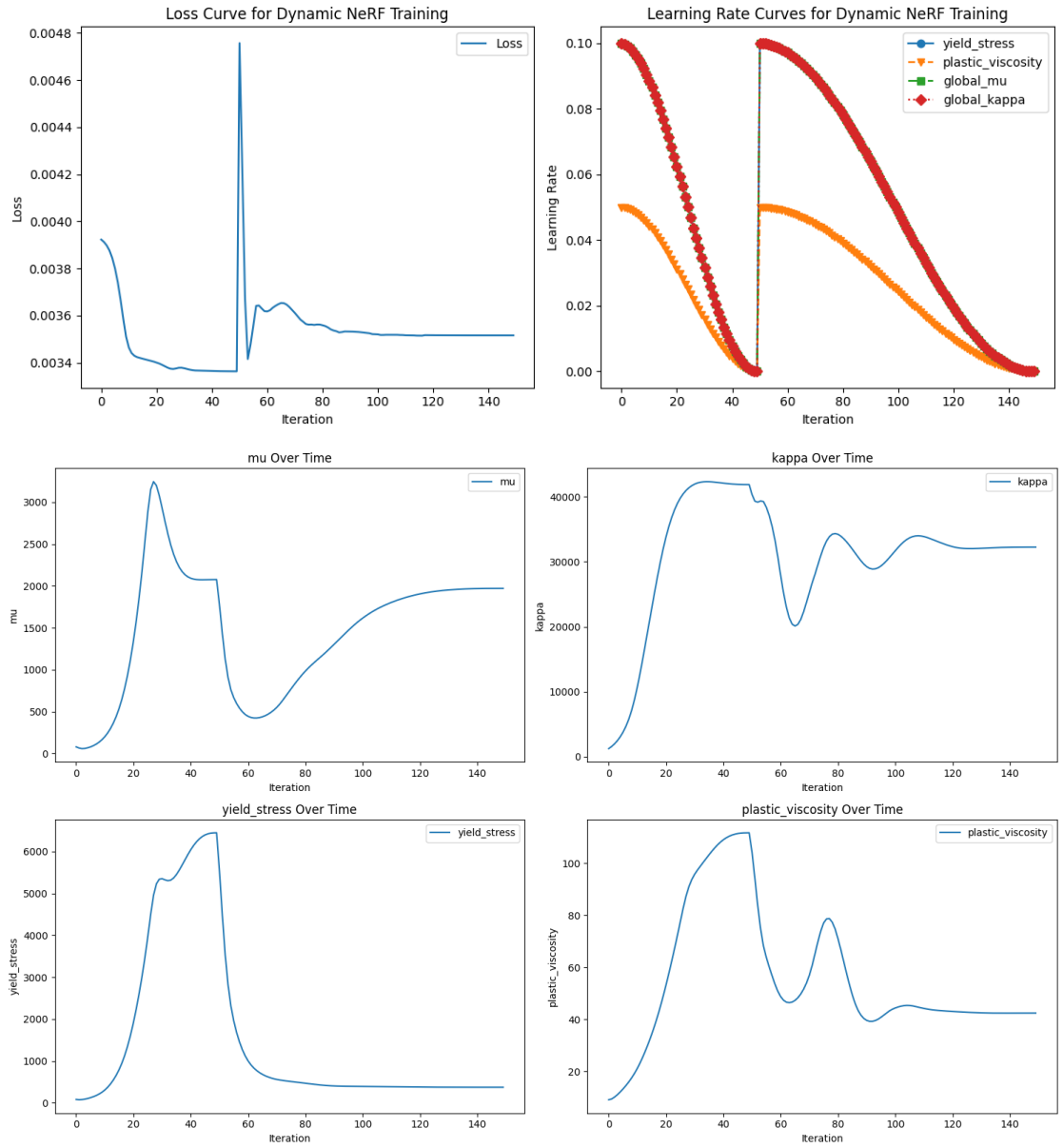


Figure A.2: Training plots for large virtual ketchup squirt: loss and learning rates (top two), material parameters (bottom four).

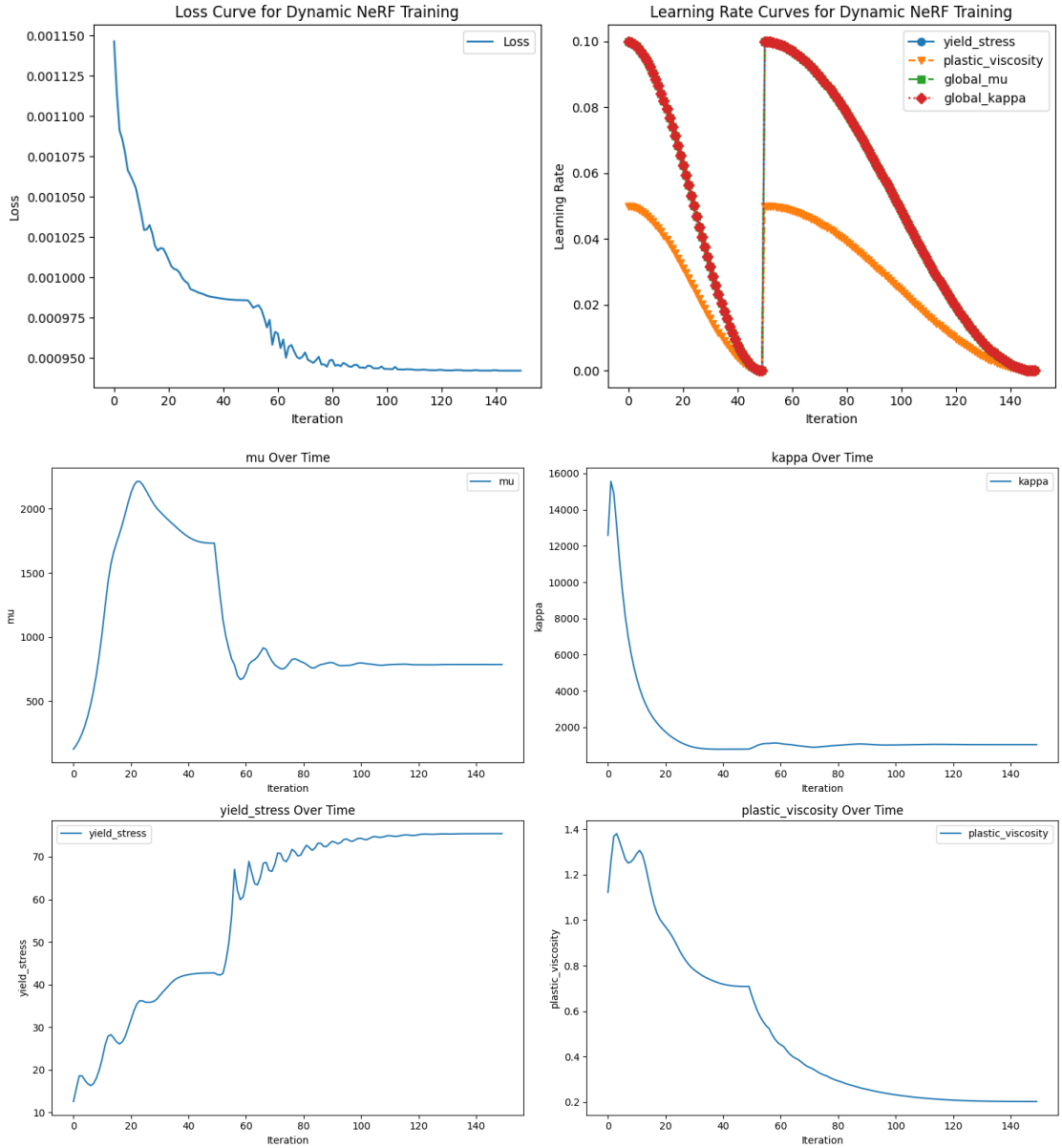


Figure A.3: Training plots for small virtual ketchup squirt: loss and learning rates (top two), material parameters (bottom four).

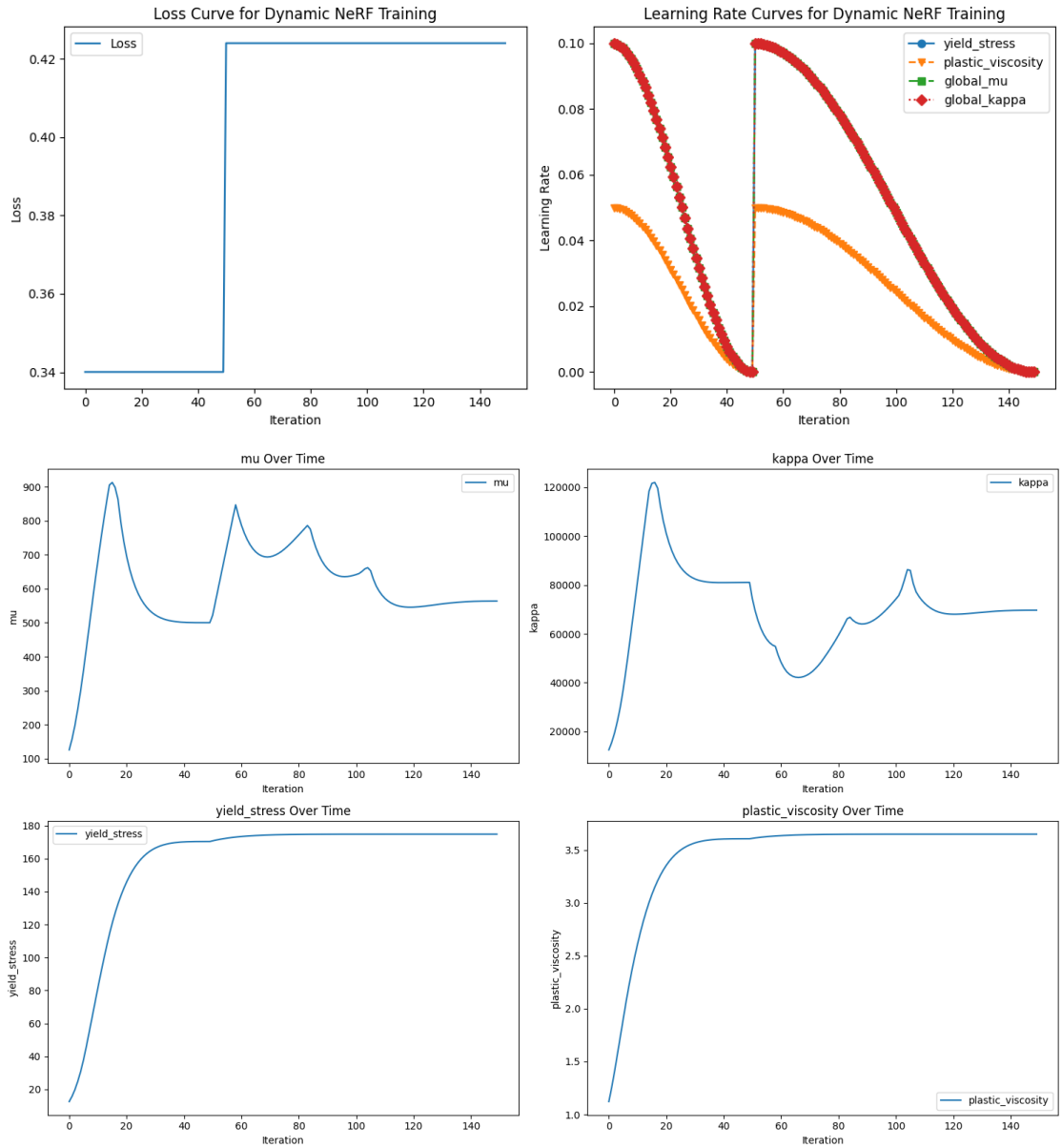


Figure A.4: Training plots for real-world ketchup cup: loss and learning rates (top two), material parameters (bottom four).

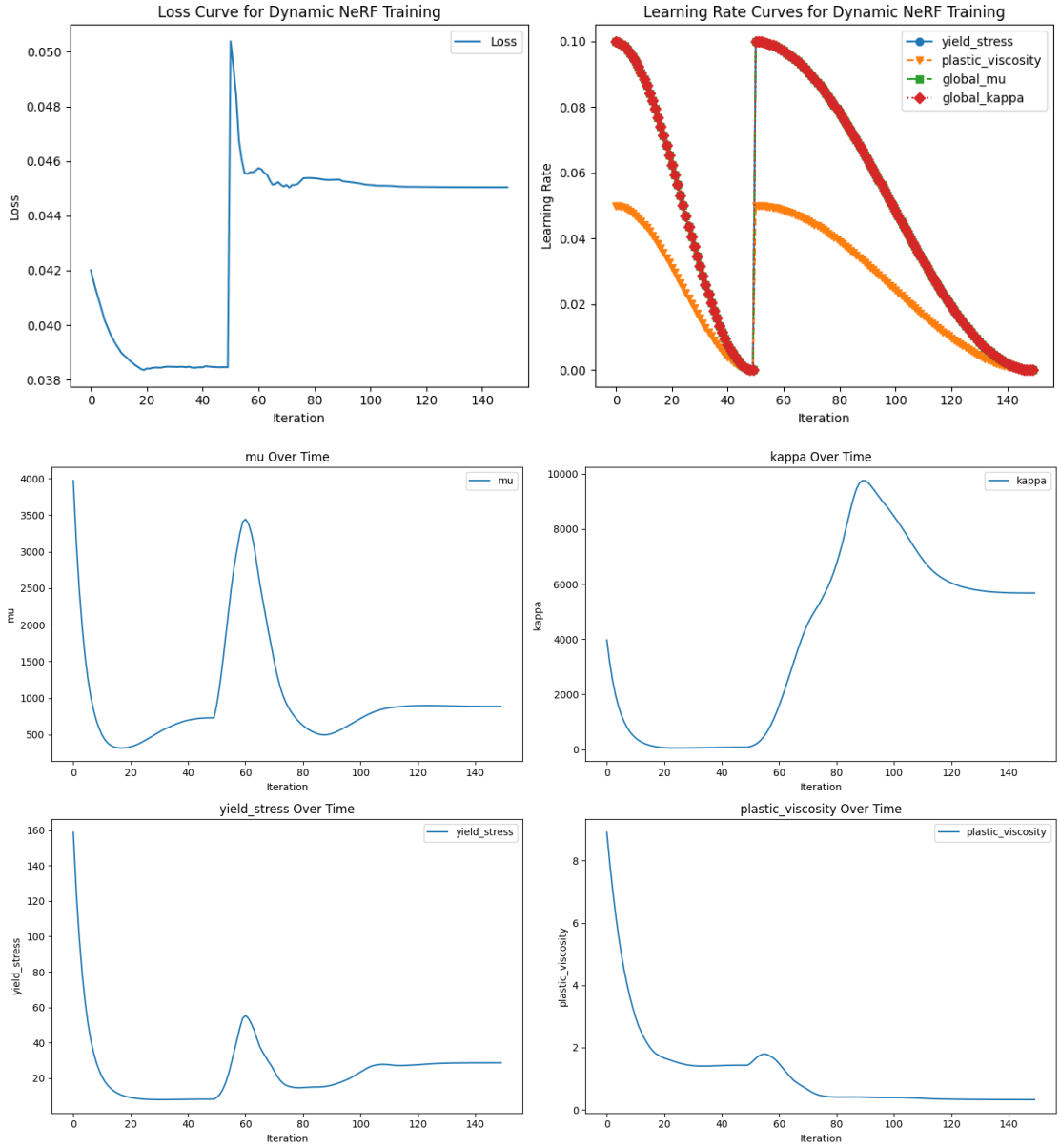


Figure A.5: Training plots for real-world ketchup squeeze: loss and learning rates (top two), material parameters (bottom four).

References

[1] P. Ma, P. Y. Chen, B. Deng, J. B. Tenenbaum, T. Du, C. Gan, and W. Matusik, *Learning neural constitutive laws from motion observations for generalizable pde dynamics*, 2023. arXiv: 2304.14369 [cs.LG].

[2] X. Li, Y.-L. Qiao, P. Y. Chen, K. M. Jatavallabhula, M. Lin, C. Jiang, and C. Gan, *Pac-nerf: Physics augmented continuum neural radiance fields for geometry-agnostic system identification*, 2023. arXiv: 2303.05512 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2303.05512.

[3] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, *Nerf: Representing scenes as neural radiance fields for view synthesis*, 2020. arXiv: 2003.08934 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2003.08934.

[4] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla, *Nerfies: Deformable neural radiance fields*, 2021. arXiv: 2011.12948 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2011.12948.

[5] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer, *D-nerf: Neural radiance fields for dynamic scenes*, 2020. arXiv: 2011.13961 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2011.13961.

[6] E. Tretschk, A. Tewari, V. Golyanik, M. Zollhöfer, C. Lassner, and C. Theobalt, *Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video*, 2021. arXiv: 2012.12247 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2012.12247.

[7] F. de Avila Belbute-Peres, K. Smith, K. Allen, J. Tenenbaum, and J. Z. Kolter, "End-to-end differentiable physics for learning and control," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31, Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/file/842424a1d0595b76ec4fa03c46e8d755-Paper.pdf.

[8] C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin, "The affine particle-in-cell method," *ACM Trans. Graph.*, vol. 34, no. 4, Jul. 2015, ISSN: 0730-0301. DOI: 10.1145/2766996. [Online]. Available: https://doi.org/10.1145/2766996.

[9] C. Jiang, C. Schroeder, J. Teran, A. Stomakhin, and A. Selle, "The material point method for simulating continuum materials," in *ACM SIGGRAPH 2016 Courses*, ser. SIGGRAPH '16, Anaheim, California: Association for Computing Machinery, 2016, ISBN: 9781450342896. DOI: 10.1145/2897826.2927348. [Online]. Available: https://doi.org/10.1145/2897826.2927348.

[10] K. Um, R. Brand, Yun, Fei, P. Holl, and N. Thuerey, *Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers*, 2021. arXiv: 2007.00016 [physics.comp-ph]. [Online]. Available: https://arxiv.org/abs/2007.00016.

[11] K. M. Jatavallabhula, M. Macklin, F. Golemo, *et al.*, *Gradsim: Differentiable simulation for system identification and visuomotor control*, 2021. arXiv: 2104.02646 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2104.02646.

[12] T. Du, K. Wu, P. Ma, S. Wah, A. Spielberg, D. Rus, and W. Matusik, "Diffpd: Differentiable projective dynamics," *ACM Trans. Graph.*, vol. 41, no. 2, Nov. 2021, ISSN: 0730-0301. DOI: 10.1145/3490168. [Online]. Available: https://doi.org/10.1145/3490168.

[13] H.-y. Chen, E. Tretschk, T. Stuyck, P. Kadlecek, L. Kavan, E. Vouga, and C. Lassner, *Virtual elastic objects*, 2022. arXiv: 2201.04623 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2201.04623.

[14] B. Wang, Y. Deng, P. Kry, U. Ascher, H. Huang, and B. Chen, *Learning elastic constitutive material and damping models*, 2020. arXiv: 1909.01875 [cs.GR]. [Online]. Available: https://arxiv.org/abs/1909.01875.

[15] X. Li, Y. Cao, M. Li, Y. Yang, C. Schroeder, and C. Jiang, "Plasticitynet: Learning to simulate metal, sand, and snow for optimization time integration," in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022. [Online]. Available: https://openreview.net/forum?id=_WqHmwoE7Ud.

[16] S. Guan, H. Deng, Y. Wang, and X. Yang, *Neurofluid: Fluid dynamics grounding with particle-driven neural radiance fields*, 2022. arXiv: 2203.01762 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2203.01762.

[17] Z. Xian, B. Zhu, Z. Xu, H.-Y. Tung, A. Torralba, K. Fragkiadaki, and C. Gan, *Fluidlab: A differentiable environment for benchmarking complex fluid manipulation*, 2023. arXiv: 2303.02346 [cs.RO]. [Online]. Available: https://arxiv.org/abs/2303.02346.

[18] W. H. Herschel and R. Bulkley, *Konsistenzmessungen von gummi-benzollösungen*, 1926. [Online]. Available: https://doi.org/10.1007/BF01432034.

[19] Y. Yue, B. Smith, C. Batty, C. Zheng, and E. Grinspun, "Continuum foam: A material point method for shear-dependent flows," *ACM Trans. Graph.*, vol. 34, no. 5, Nov. 2015, ISSN: 0730-0301. DOI: 10.1145/2751541. [Online]. Available: https://doi.org/10.1145/2751541.

[20] T. Takahashi and M. C. Lin, "Video-guided real-to-virtual parameter transfer for viscous fluids," *ACM Trans. Graph.*, vol. 38, no. 6, Nov. 2019, ISSN: 0730-0301. DOI: 10.1145/3355089.3356551. [Online]. Available: https://doi.org/10.1145/3355089.3356551.

[21] M. Hamamichi, K. Nagasawa, M. Okada, R. Seto, and Y. Yue, "Non-newtonian virheometry via similarity analysis," *ACM Trans. Graph.*, vol. 42, no. 6, Dec. 2023, ISSN: 0730-0301. DOI: 10.1145/3618310. [Online]. Available: https://doi.org/10.1145/3618310.

[22] C. Sun, M. Sun, and H.-T. Chen, *Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction*, 2022. arXiv: 2111.11215 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2111.11215.

[23] S. Lin, A. Ryabtsev, S. Sengupta, B. Curless, S. Seitz, and I. Kemelmacher-Shlizerman, *Real-time high-resolution background matting*, 2020. arXiv: 2012.07810 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2012.07810.

[24] C. White and J. M. Lees, "Yield stress prediction from 3d reconstruction of fresh concrete slump," *Cement and Concrete Research*, vol. 174, p. 107 331, 2023, ISSN: 0008-8846. DOI: https://doi.org/10.1016/j.cemconres.2023.107331. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0008884623002454.

[25] *The stanford 3d scanning repository*. [Online]. Available: https://graphics.stanford.edu/data/3Dscanrep/.

[26] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, *3d gaussian splatting for real-time radiance field rendering*, 2023. arXiv: 2308.04079 [cs.GR].

[27] I. Loshchilov and F. Hutter, *Sgdr: Stochastic gradient descent with warm restarts*, 2017. arXiv: 1608.03983 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1608.03983.