

Physics-Informed 3D Gaussian Splatting for Rainy Scene Simulation

Elena-Oana Milchi

Physics-Informed 3D Gaussian Splatting for Rainy Scene Simulation

by

Elena-Oana Milchi

Advisor:	Dr. M. Weinmann & Dr. N. Saptarshi Sinha
Graduation committee:	Dr. M. Weinmann & Dr. N. Saptarshi Sinha & Dr. X. Zhang
Research Group:	Computer Graphics & Visualization
Project Duration:	November, 2025 - June, 2026
Faculty:	Electrical Engineering, Mathematics and Computer Science

Acknowledgements

This thesis represents not just the finalization of my master studies, but rather an end to seventeen years of academic pursuit, of continuous (and, at times, strenuous) learning and development. I dedicate this report to my support system, whose encouragement carried me here, and to my younger self, who would hardly believe how far we have come.

I would like to extend my sincere gratitude to my supervisors, Dr. Michael Weinmann and Dr. Neil Saptarshi Sinha, whose guidance and support have been invaluable not just to my progress throughout the thesis project, but also to my personal development as a researcher. I will always admire their dedication to the craft and their consistent pursuit of improvement, and will seek to approach my future career with the same amount of passion as them. I also send my warmest regards to the Computer Graphics and Vision department at TU Delft, who has inspired me, welcomed my ideas and provided valuable feedback, whose vast knowledge and work I admire deeply.

I extend my warmest thanks to my friends, who have consistently stood by me and patiently endured my tangents on cubes, chairs, and hotdogs.

Finally, I am deeply grateful to my parents, whose unwavering support is the primary reason I was able to undertake this research at all.

*Elena-Oana Milchi
Delft, June 2026*

Abstract

Simulating the infiltration of rainwater into solid materials is of broad interest across multiple fields, such as agriculture, hydrology, urban planning, and landslide prevention, yet most rainwater infiltration models rely on numerical simulation and offer little support for interactive visualization. Recent advances in 3D scene representation, particularly 3D Gaussian Splatting (3DGS), enable high-quality and photorealistic scene captures that promise an inherent integration with particle-based physics simulation engines. However, existing 3DGS-based weather frameworks emphasize appearance modeling and do not address the interaction between rainwater and materials within a 3D scene.

In this thesis, we present a proof of concept tackling the physics-informed simulation of rainwater infiltration into solids represented by 3D Gaussian Splatting. We build on the FeatureSplatting pipeline and extend it from single-object editing to an interactive visualization of rainwater infiltration into solid media. Our approach comprises three components: (1) a preprocessing pipeline that prunes Gaussian artifacts and applies point cloud infilling strategies to transpose a 3DGS representation into particle space, (2) an extension to the MPM solver that models rainwater infiltration as the transfer of fluid between macroscopic porous particles, and (3) functionality updating 3D Gaussian appearance parameters driven by per-particle saturation. We evaluate the method on synthetic 2D and 3D scenarios and show that it reproduces expected physical behavior, as well as analyze in depth the main factors that govern the rate of fluid spread.

Contents

Acknowledgements	i
Abstract	ii
1 Introduction	1
2 Related Work	3
2.1 Learning-based scene representation	3
2.2 Physics-Informed Scene Representations	3
2.3 Fluid dynamics and Gaussian Splatting	4
2.4 Physically-grounded fluid-solid coupling simulations	4
3 Background	6
3.1 3D Gaussian Splatting	6
3.2 FeatureSplatting	7
3.3 The Material Point Method (MPM)	7
4 Methodology	10
4.1 3D Gaussian to particle representation	10
4.1.1 3D Gaussian pruning	11
4.1.2 Particle densification	11
4.2 MPM simulation	12
4.2.1 Simulation initialization	12
4.2.2 Particle to Grid (P2G)	12
4.2.3 Grid processing	13
4.2.4 Grid to Particle (G2P)	14
4.3 Appearance parameter update	14
5 Experimental Setup and Results	16
5.1 Input data and preprocessing	16
5.1.1 Horizontal 2D Grid Scene	16
5.1.2 Synthetic Sponge Scenes	17
5.2 MPM simulation	20
5.2.1 Horizontal 2D Grid	20
5.2.2 Synthetic cube	24
5.3 Appearance parameter update	25
6 Discussion	30
6.1 Discussion of results	30
6.2 Limitations	32
6.3 Reflection on research in a broader societal context	33
7 Conclusion and Future Work	34
References	35
A Reflection on use of Generative AI	39

1

Introduction

Rainwater infiltration is a topic of great interest in multiple areas, including agriculture, hydrology [1], urban planning [2] and landslide prevention [3]. Simulation of soil-water relationships during rainfall can yield insights into rainwater infiltration in urban as well as natural areas. On the one hand, rainfall runoff on concrete roads can cause accumulations leading to flooding [2], making simulations essential in the design of drainage systems for cities and roads [2], [4]. Most rainwater infiltration models rely on numerical simulation, however, thus lacking the possibility for the interactive exploration and visualization of urban or natural scenes.

Recent advancements in computer vision, specifically in 3D scene representation, have achieved the high-quality scene captures, enabling a photorealistic visualization. Current discourse in the literature in that sense follows two distinct directions: either using MLPs to obtain an implicit scene representation via Neural Radiance Fields [5], or obtaining an explicit representation via 3D Gaussian Splatting (3DGS) [6]. Whereas both methods achieve photorealistic view synthesis, 3DGS achieves similar render quality with less training time. Moreover, the resulting output from 3DGS is highly compatible with particle-based physics simulations.

Following the advent of 3DGS, multiple studies have emerged that apply this novel rendering method in physics simulations. In most cases, the obtained Gaussian Splats are preprocessed, fed into a physics engine, then postprocessed based on the engine's result. The employed physics engines can be either mesh-based [7], or particle-based [8], [9]. For the modeling of adverse weather specifically, two 3DGS-based frameworks have emerged. On the one hand, RainyGS [10] combines a 3DGS representation with the use of height maps and ray tracing in order to augment photorealistic scenes with adverse weather conditions. The movement of water on the surface is dictated by shallow water equations [10]. On the other hand, Let it Snow! [11] employs a ready-made particle-based physics engine to model the interactions between multiple particles. The main focus of this study is on particle appearance, allowing for the simulation of multiple weather phenomena such as rain, snow and sandstorms. Both RainyGS [10] and Let it Snow! [11] handle the accumulation of water onto solid surfaces, yet both are geared toward appearance modeling. Neither method focuses on the interaction between rainwater and the input scene from a physics perspective.

The purpose of this study is to investigate how to simulate realistic patterns of fluid infiltration into real-life solid materials. To this end, we aim to answer the following research questions:

1. RQ1: How can we simulate the interaction of rainwater with solid materials in a scene represented by 3D Gaussian Splatting?
2. RQ2: In what ways can we augment the underlying 3D Gaussian representation with material-aware infiltration and saturation characteristics?
3. RQ3: How can we gain evidence on the potential of the solution developed for the interaction between rain and scene materials?

4. RQ4: In what ways do the qualities of the underlying 3DGS representation impact the result of the rain simulation?

When considering the simulation of fluid dynamics, three main approaches emerge from the literature. One can either follow the Eulerian approach, where data on fluid measurements is stored at discrete points in a grid, or the Lagrangian perspective, where fluid is represented as a set of particles interacting with each other [12]. Subsequently, hybrid methods have emerged building on the strengths of both approaches. A rather recent yet highly capable method has been the Material Point Method (MPM) [13]. This simulation technique takes as input a set of particles, uses this data to construct a grid where velocities across the scenes are computed, then updates the particles based on the results obtained from the grid computation [13]. The MPM has been extended to handle solid matter, as well as incompressible fluids and snow [13]. Moreover, it has proven quite versatile especially in large-scale simulations, such as for the simulation of landslides generated by the infiltration of water into the soil [14].

Due to its particle-based nature, the MPM integrates seamlessly with 3DGS pipelines, as 3D Gaussian functions can intuitively be transposed in particle space. This has in fact been achieved by FeatureSplatting [9], which allows for selective Gaussian editing and physics simulation. It leverages vision-language models to segment objects within a scene based on similarity scores, then initializes an MPM simulation with only the selected objects. By coupling an MPM-based physics engine with a 3DGS representation of a scene, the framework proves useful for the purpose of this study.

In this thesis, we research the possibilities for simulating the physics-based interaction between rain and the materials in a 3D scene, with a focus on the infiltration of rainwater into solids. To the best of our knowledge, this task has not yet been approached in current literature. We build our work on top of FeatureSplatting [9], and extend it to a proof of concept that can quantify the infiltration of rainwater into stationary particles. Taking inspiration from [15], we treat solid objects as a collection of macroscopic particles, and extend FeatureSplatting's underlying MPM simulation to handle the transfer of fluids between porous particles, aiming to model their infiltration and propagation throughout materials in the scene. The simulation returns a saturation value per particle, which is in turn used to inform appearance changes in associated 3D Gaussians. Our contribution is thus threefold:

1. A preprocessing pipeline that takes as input a 3DGS representation of a scene, prunes artifacts and applies appropriate infilling strategies to transpose the representation into particle space.
2. An extension to a pre-existing MPM simulation that treats rainwater infiltration as a the inter-particle transfer of fluid mass between macroscopic water-retaining particles, allowing for handling different materials as well. By measuring fluid mass at multiple points within an Eulerian grid, the method obtains smooth patterns of fluid transfer through solid media.
3. A modification of 3D Gaussian appearance parameters based on per-particle saturation values obtained in the MPM simulation.

2

Related Work

In this section, we provide a brief survey of existing methods used in 3D scene representation, as well as their integration with physics engines to simulate fluids alongside their interactions with solid materials. We begin with a review of methods used in scene representation, from the early stereo approaches to state-of-the-art methods relying on deep learning. We then explore how neural methods used in novel view synthesis are augmented with physics priors to produce realistic simulations, with a focus on the applications of simulations in modern Gaussian Splatting pipelines. Lastly, we highlight the different types of physics engines most often employed in the simulation of fluids.

2.1. Learning-based scene representation

Geometry representation and scene reconstruction have been long-standing challenges in the computer graphics and vision communities. A well known method in the field with numerous large-scale and industrial applications has been Structure From Motion (SFM) [16]. The method takes as input a set of images, and infers corresponding camera parameters per scene, as well as a sparse point cloud [16]. Modern Multi-View Stereo (MVS) pipelines often rely on these output camera parameters to reconstruct 3D geometry from a set of input images by deriving stereo correspondences [16].

As deep learning grew in popularity, multiple methods emerged for scene modelling in the form of voxels, point clouds or meshes, as well as implicit representations [17]. As such, Mildenhall et. al.'s Neural Radiance Fields (NeRF) [5], model 3D scenes implicitly by training an MLP for directional color prediction. This MLP takes as input a 3D position and view direction, and outputs a density value as well as a color vector [5]. NeRF is remarkably the first continuous neural scene representation able to produce photorealistic novel renders from a set of input images [5]. However, because the underlying MLP must be queried across multiple points per viewing direction, the training times of the original implementation have proven to be rather expensive.

To address the inefficient training times in early NeRF implementations, 3D Gaussian Splatting (3DGS) trains an explicit scene representation based on 3D Gaussian primitives [6]. The algorithm takes as input a series of images, computes correspondences between them to initialize a set of 3D Gaussian Functions, then optimizes their position, covariance and appearance parameters via gradient descent [6]. This representation is faster to render and, due to its explicit nature, better suited for scene editing.

2.2. Physics-Informed Scene Representations

Following the advent of radiance fields, multiple contemporary methods aimed to leverage them to model physics simulations. Two strategies of integrating physical priors in NeRFs are employing Physics-Informed Neural Networks (PINNs) [18] and particle sampling of NeRFs augmented with physical attributes [19], [20]. For instance, PINNs have been applied to learn velocity fields which, combined with radiance fields, enabled the rendering of fluids such as smoke [18]. On the other hand, particle sampling of NeRFs has led to Lagrangian [19] as well as hybrid Lagrangian-Eulerian [20] representations of materials. DEM-NeRF [21] combines particle sampling with PINNs to model hyperelasticity

in solid objects. The prominence of particle representations raises the question whether Gaussian Splatting-based pipelines would be more intuitive for physics integration instead.

Most studies that integrate 3DGS approaches to model physical phenomena embed a physics engine into their pipelines - either prior existing or custom-made. Borycki et. al. [7] adapt a 3DGS-based representation of a scene to a black-box physics engine by encoding each Gaussian as three triangle vertices. The algorithm is able to produce high quality simulations without any modifications to the physics engine [7]. [22] makes use of a pre-existing differentiable physics engine to learn elasticity parameters given a 3DGS-based scene representation, producing novel interactions involving elastic objects at faster inference times. On the other hand, PhysGaussian [8] transforms the Gaussians representing the scene as a particle cloud that is deformed over time. The method captures the kinematic movements and deformation of elastic objects, metals and granular media using a custom-made MPM [8].

In contrast, there are also a few 3DGS-based works for modeling physics phenomena that do not employ a physics engine. In particular, Physdreamer [23] bypasses the need for one, instead relying on video generative models to simulate object dynamics.

2.3. Fluid dynamics and Gaussian Splatting

Many 3DGS-based methods synthesize fluids by means of Lagrangian-based methods. Some initialize Lagrangian fluid particles, then optimize them in accordance to physics-informed losses targeting fluid vorticity or visual and geometric consistency [24], [25]. Gaussian Splashing uses a Lagrangian particle simulator to capture the interaction between solids and fluids [26]. In contrast, FluidGS [27] takes a hybrid Lagrangian-Eulerian approach. The method models smoke by modeling particles with 3DGS, then transferring them to an Eulerian grid, where a PINN learns the velocity field of the fluid using Navier-Stokes equations. An alternative strategy [28] is the dynamic creation, evolution and pruning of Gaussians directly according to a physics-informed loss, thus approximating time-dependent PDEs that model fluid dynamics.

Recently, two distinct 3DGS-based methods have emerged to simulate adverse weather. The first, RainyGS [10], represents raindrops as anisotropic 3D Gaussians, and the resulting water accumulation on the ground is dictated by shallow water equations. The rasterization approach of 3DGS is then combined with ray tracing to render water reflectivity. The second method, Let it snow! [11], takes a different approach, mapping Gaussian kernels to particles and modelling complex dynamics with the Material Point Method (MPM). Upon mapping the particles back to Gaussians, appearance parameters are computed manually for simpler appearances, and using a 3DGS diffusion model for complex parameters [11]. Both methods are geared towards appearance modelling, lacking a concrete model of interactions between the rain and the scene in a physics-based sense.

2.4. Physically-grounded fluid-solid coupling simulations

Fluid solvers have been a significant research area in the field of computer graphics. Two approaches are most widely used to track the motion of fluids: the Lagrangian approach, whereby each fluid point is defined by a position and velocity, and the Eulerian approach, which stores characteristics of the volume of fluids at specific points in space [12]. As opposed to the Lagrangian perspective, the Eulerian representation allows for a differentiable rendition of fluids [12], [25].

An example of Lagrangian fluid simulation is the Smooth Particle Hydrodynamics (SPH) method [29]. Originally applied in astrophysics to study the evolution of asymmetric systems [29], the method has been deemed to offer satisfactory approximations of fluid dynamics equations [30]. SPH treats a fluid as a series of particles and derives continuous properties from this discrete set via interpolation [15], [30]. In contrast, the Marker-and-Cell method [31] was the first to introduce a computational grid as an Eulerian representation of the input space. This grid provides a volumetric scene representation in terms of three-dimensional cells which tracked and computed pressure and velocity [31], which facilitated the numerical approximation of their spatial derivatives [12].

The Material Point Method (MPM) has emerged to combine the advantages of both Lagrangian and Eulerian representations [13]. The method assumes a particle-based representation of the scene, then

transfers it to an Eulerian grid where particle velocities are computed, then advects the particles based on velocities transferred from the grid [13]. The method has been found to efficiently model large-scale deformations, thus being often employed in the field to simulate landslides and the infiltration of water into soil [14]. To model fluid-solid coupling in MPM simulations, fluid and solid particles are often treated separately in different phases [32]. Furthermore, multi-phase MPM simulations can be formulated as either single-point or multi-point. Whereas single point methods incorporate information on soil and pore water content in a single particle, multi-point methods separate these properties into different particle sets [14].

In recent years, as deep learning grew in popularity, multiple methods began leveraging neural networks for fluid simulation. For instance, CNN models with physics priors integrated into their loss functions were proven to model incompressible fluid dynamics by solving the Navier-Stokes and damped wave equations [33], [34]. Similarly, [35] trains a U-Net to learn velocity components across a Marker-and-Cell grid, which enables modeling incompressible fluids in 3D.

3

Background

This chapter provides a detailed discussion of all of the frameworks, methods and technologies we build upon in our methodology. We begin by highlighting the basics of 3D Gaussian Splatting [6], namely the algorithm producing the input to our physics simulation. We then proceed to discuss FeatureSplatting [9], the study we build upon in our methodology. Lastly, we detail the inner workings of the physics engine extended in this study, which is based on the Material Point Method [13].

3.1. 3D Gaussian Splatting

In recent years, 3D Gaussian Splatting (3DGS) [6] has emerged as a promising fluid rendering method due to its capability to create high-quality scene representations in real time [25]. The optimization begins from a sparse point cloud, which is obtained either by Structure From Motion [36], [37] or by random point sampling. These points inform the initialization of 3D Gaussians defined by a position or mean μ , a covariance matrix Σ , which determines the Gaussian’s rotation and scaling, and opacity α [6]. For one point x in the input point cloud, the modeled Gaussian function is defined below:

$$G(x) = e^{-\frac{1}{2}(x)^T \Sigma^{-1} x} \quad (3.1)$$

Where Σ represents the covariance matrix, which can be expressed in terms of a scaling and rotation matrix as follows:

$$\Sigma = R S S^T R^T \quad (3.2)$$

The optimization process accounts for all Gaussian parameters (i.e. position, covariance, transparency and spherical harmonics) via Stochastic Gradient Descent, using the following objective function:

$$\mathcal{L} = (1 - \lambda) \mathcal{L}_1 + \lambda \mathcal{L}_{D-SSIM} \quad (3.3)$$

Where $\mathcal{L}_1$ is the per-pixel absolute difference between the rendered image and the ground truth image, and $\mathcal{L}_{D-SSIM}$ represents the Differential Structural Similarity Index (D-SSIM). The loss is regularized by the parameter λ . Besides this optimization, the method introduces an adaptive control process to account for under- or -ver-reconstructed areas. This process clones the Gaussians in areas where geometric features are insufficiently modeled, and splits Gaussians in areas where a single large splat represents small scale geometry [6]. Moreover, geometry with an opacity under a certain threshold is removed from the scene.

To render the 3D Gaussian splat onto a 2D screen, Kerbl et. al [6] have implemented a differentiable rasterizer. The rasterizer projects Gaussians from 3D onto a 2D screen space, then sorts them based on their depth. The final color of a pixel in a rendered scene, based on all depth-sorted Gaussians that overlap it ($i \in N$) is dictated by the following formula:

$$\hat{C} = \sum_{i \in N} c_i \cdot \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) \quad (3.4)$$

Where c_i is the view-dependent color of Gaussian i , α_i is its opacity, and the product term represents the transmittance accumulated over preceding Gaussians. Overall, the method is able to obtain an editable 3D scene representation within a fast training time.

3.2. FeatureSplatting

FeatureSplatting [9] is a versatile 3DGS framework initially introduced for language-guided and physics-based scene editing. It allows for segmenting objects in a photorealistic 3D scene and either directly translating and rotating them, or running physics simulations on them. Each 3D Gaussian in the scene is enhanced with a vector of features, whose splats are compared against per-part CLIP features during supervised training. Thus, for a set of Gaussians with feature vector f_i , view-dependent color c_i and opacity α_i , the rendered color and visual features are computed as:

$$\{\hat{F}, \hat{C}\} = \sum_{i \in N} \{f_i, c_i\} \cdot \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) \quad (3.5)$$

The rendered features are regularized via a small neural network, which produces two feature maps, i.e.:

$$\hat{F}_C, \hat{F}_D = MLP(\hat{F}) \quad (3.6)$$

Where $\hat{F}_C$ is trained using a SAM-enhanced [38] feature map (M) as highlighted in Equation 3.7, and $\hat{F}_D$ is similarly trained using a DINOv2 [39] feature map as a target. The two feature maps are jointly optimized in the loss function:

$$w = MAP(M, F_C) = \frac{\sum_{i \in F_C} M(i) \cdot \frac{F_C(i)}{\|F_C(i)\|}}{\sum_{i \in F_C} M(i)} \quad (3.7)$$

When a multi-material object in the scene is selected, FeatureSplatting employs the features stored in each Gaussian to predict its material [9]. The model is then able to simulate rigid, elastic, granular or liquid particles using a customized Material Point Method based on the Taichi engine [40]. Volume preservation is handled by sampling additional particles along the surface, as well as within, the simulated objects. This aspect in particular is of high relevance to simulating rainwater seepage. Figure 3.1 visualizes FeatureSplatting’s potential of rendering collapses of granular materials on the one hand, and elasticity on the other hand.

Due to its integration of a physics engine in a 3DGS pipeline, which enables the direct editing of 3D Gaussians in the scene based on the simulation results, FeatureSplatting [9] represents a strong basis to our study.

3.3. The Material Point Method (MPM)

The Material point Method has been initially introduced in the simulation of snow, and has since been extended to handle multiple materials, including fluids, viscoelastic and elastoplastic materials [32]. Most notably, the method has proved particularly capable of simulating large-scale interactions between fluids and solids. The MPM takes as input a range of particles with pre-set properties such as position, mass, volume, and velocity, computes new velocities in one simulation step and advects the particles accordingly [13]. In contrast to purely Lagrangian methods such as Smoothed Particle Hydrodynamics (SPH), the MPM computes new velocity gradients on a grid, leading to speedups in computation. Moreover, maintaining the particle-based representation leads to more accurate results than purely Eulerian methods.

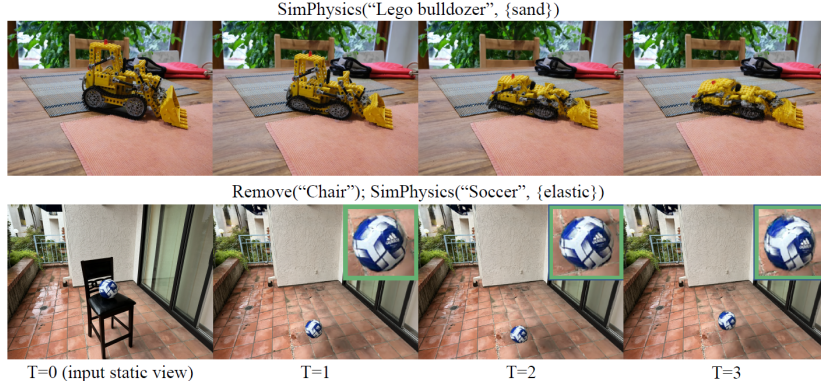


Figure 3.1: Physics simulation renderings given photorealistic scenes produced by FeatureSplatting. The first scene renders the collapse of solid materials turned granular, and the second visualizes the elasticity of a bouncing ball. Image taken from [9]

The MPM can be thought to comprise four different steps: The transfer from particles to an Eulerian Grid, the update of velocities on the grid, the transfer from the grid to the particles, and particle advection. The traditional MPM aims to only compute particle motion based on Newtonian laws [13]. Many extensions geared for fluid simulation will handle fluids and solids separately [32]. The MPM grid usually handles velocity, momentum, or pressure transfers between particles [13].

In the Particle-to-Grid (P2G) step, the Affine-Particle-In-Cell method (APIC) is often employed to compute transfers of mass and velocity [13]. For a set of particles p with mass m_p , velocity v_p and position x_p , their contributions to the mass m_n and velocity v_n of the corresponding grid node n are calculated in Equations 3.8 and 3.9.

$$m_n = \sum_p w_{np} m_p \quad (3.8)$$

$$m_n v_n = \sum_p w_{np} m_p (v_p + B_p (D_p)^{-1} (x_n - x_p)) \quad (3.9)$$

Where w_{np} represents the weight given to particle p 's contribution to grid node n , B_p is an affine matrix stored per particle. Moreover, D_p , which is analogous to an inertia tensor, is computed as:

$$D_p = \sum_i w_{np} (x_n - x_p) (x_n - x_p)^T \quad (3.10)$$

On the grid, velocities are updated according to nodal forces f_i and the particle deformation gradient is computed [13]. In the weak form MPM, f_i can be computed as a function of the nodal volume V_p^n and Cauchy stress tensor σ_p^n according to:

$$f_i^n = - \sum_p V_p^n \sigma_p^n \nabla w_{ip}^n \quad (3.11)$$

The nodal force is then used to update the grid node position and velocity according to:

$$x_i^{n+1} = x_i^n + \Delta t v_i^{n+1} \quad (3.12)$$

$$v_i^{n+1} = v_i^n + \Delta t f_i(x_i^n) / m_i \quad (3.13)$$

Once grid velocities have been computed, the information is transferred back into particle space in the Grid-to-Particle (G2P) step. This is achieved following the APIC method according to:

$$v_p = \sum_i w_{ip} v_i \quad (3.14)$$

$$B_p = \sum_i w_{ip} v_i (x_i - x_p)^T \quad (3.15)$$

Finally, once particle velocities are computed in the G2P step, particles are advected according to their velocities and velocity gradients. Their position will change, thus concluding one step in an MPM simulation.

4

Methodology

The purpose of this thesis is to develop a physics-informed rain simulation which takes as input photorealistic scenes. The main aspect under study is the dispersion of water into the ground and into objects of varying materials. It is important to note that the method is an extension of the NeRFStudio version of FeatureSplatting [9]. As such, our pipeline relies on FeatureSplatting’s distilled features associated with each Gaussian, and extends their physics simulation from a single object to an entire scene.

We begin with a trained 3D Gaussian representation, which we can intuitively transpose to a particle-based representation. This allows for great compatibility with particle-based and hybrid physics simulation models. We consider the scene as a space of stationary particles, to which we add dynamic rain particles. The method will keep track of the mass of fluid being transferred from rain into solid matter via a customised Material Point Method (MPM). Subsequently, saturation values per particle obtained from the MPM simulation will be used to update appearance parameters of the 3D Gaussians representing the scene. Our methodology is visualized in Figure 4.1 and aims to answer RQ1 and RQ2 (See Chapter 1).

We thus distinguish three main steps in our method: (1) obtaining a point cloud from a 3D Gaussian representation, (2) running a physics simulation to determine saturation levels among stationary particles in the scene, and (3) leveraging the information from this physics simulation in a post-processing step that updates the existing 3D Gaussians.

4.1. 3D Gaussian to particle representation

We begin with a pre-trained 3D Gaussian representation. The NeRFStudio version of FeatureSplatting [9] allows the user to input an object in the scene to be segmented and used in a physics simulation.

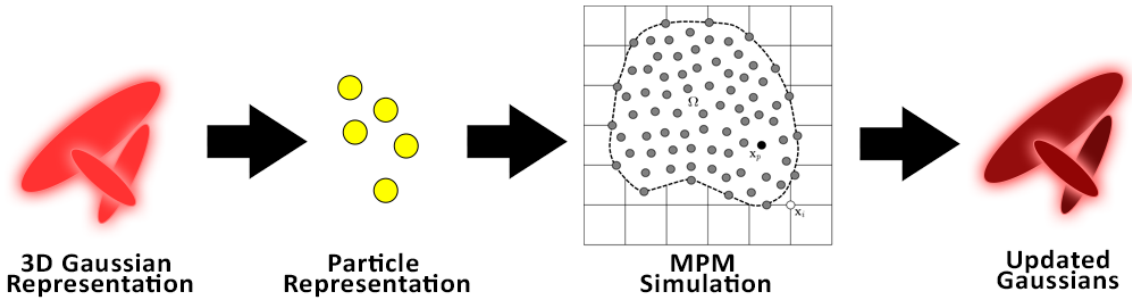


Figure 4.1: The pipeline of our proposed method. A trained 3D Gaussian [6] representation is taken as input. This representation is pruned and transformed into a particle representation, which is used as input to a custom Material Point Method (MPM) [13] simulation which keeps track of particle saturation. This value is then used to update the appearance of existing Gaussians in the scene.

Based on the distilled SAMCLIP features associated with each Gaussian, the framework retains a bounding box around the formation that most accurately represents the selected object. Since we wish to extend the simulation domain to the entire scene, this bounding box is re-computed. Firstly, using FeatureSplatting’s [9] ground segmentation functionality, which segments Gaussians in the scene based on the text prompt ‘floor’, we obtain the coordinates of the bounding box representing the ground plane. Then, we allow the user to specify a height for the box that should represent the entire scene. Only the Gaussians within the specified space are taken into account for the simulation. It is important to note that this method works best on flat terrain and might be unreliable for slope detection.

In order to accurately represent the input scene in the MPM simulation, two additional pre-processing steps need to be undertaken, Firstly, any Gaussian artifacts remaining from training must be pruned. Secondly, to ensure that the representation is dense enough for fluid to properly propagate through it, we must add support particles to each component of the scene separately.

4.1.1. 3D Gaussian pruning

In many cases, depending on the input images, the Gaussian representation is not fully accurate and contains artifacts. The process of pruning these artifacts is carried out in multiple steps, each of them taking into account different properties of the 3D Gaussians. Firstly, we prune the top- $x\%$ most transparent and largest Gaussians in the scene, where Gaussian size is measured as the greatest length across all three dimensions. The parameter x is configurable and is likely to vary per scene. The final representation must also be cohesive in terms of semantic meaning and spatial proximity. Thus, we run the clustering algorithm DBSCAN [41] twice on parameters of remaining Gaussians above ground level. We first cluster Gaussians according to their distilled features using cosine similarity as a distance metric, then according to their means using the euclidean distance metric. Gaussians that are not assigned to any cluster are considered outliers and excluded from the simulation.

4.1.2. Particle densification

Measuring the amount of fluid absorbed by the scene as a result of rain requires a volumetric representation of the scene. As we lack such a representation, we must add support particles to avoid inaccuracies such as raindrops passing through solid matter, as well as have a more fine-grained simulation.

Since the ground is assumed to be a planar surface, its densification algorithm is different from that of separate objects in the scene. We first sample a series of positions in the XZ plane (we treat the Y-axis as pointing vertically upward) in low density areas. We do this by randomly sampling particles from the ground plane weighted by their mean distance to their k -nearest neighbors, then adding a perturbation. Then, we train a linear interpolator on the XZ coordinates of all particles in the ground plane, with the target being the particles’ Y-coordinate. We apply this interpolator on the sampled XZ coordinates to approximate a realistic height.

Infilling separate objects first requires obtaining a segmentation of the scene. This is achieved by using DBSCAN [41] to cluster the pruned Gaussians into clusters with similar distilled features, then adding support particles by linear interpolation from the center of the object to each current particle, according to:

$$P_{S_{i,k}}^{\hat{}} = \frac{(K - k)c + kp_i}{K}, i \in \{1, \dots, N\}, k \in \{1, \dots, K\} \quad (4.1)$$

$$\hat{P}_S = P_S \cup \{P_{S_{i,k}}^{\hat{}} | i \in \{1, \dots, N\}, k \in \{1, \dots, K\}\} \quad (4.2)$$

Where c represents the object center and K is a configurable number of support particles to add per existing particle. $P_{S_{i,k}}^{\hat{}}$ is the set of support particles added for a single particle in an object, and $\hat{P}_S$ represents the final positions of all particles that are associated with a single object. To reduce memory load, only a random fraction of $\hat{P}_S$ is included in the simulation.

Note that this interpolation process will return unsatisfactory results for planar surfaces, which are usually represented by fewer, larger Gaussians. In such cases, some of the existing Gaussians associated

with these objects are erroneously removed in the pruning process. To account for planar surfaces, we use a Nearest Neighbor Interpolator that predicts a particle's Z-coordinate given its X and Y coordinates. The interpolator is trained on all particles belonging to the object, then given a dense grid in the X and Y coordinates to predict the Z-coordinate for. A fraction of these predictions is included as support particles in the simulation.

The final set of particles to be included in the simulation, which we define as P , can be expressed as a union between the set of ground particles, P_G , and all particle sets corresponding to identified objects, P_S :

$$P = \hat{P}_G \cup \{\hat{P}_S | S \in O\} \quad (4.3)$$

4.2. MPM simulation

The goal of this thesis is to develop a simple simulation which can simultaneously compute the transfer of water contents between fluid and solid particles and produce a simple parameter that would allow for visualizing patterns of fluid transfer in the 3D Gaussian space. We draw inspiration from [15], which assume porous particles to be macroscopic, and model particle saturation as a function of the contained mass of fluid. This saturation parameter will be associated with each Gaussian in the scene that has been selected for simulation and inform its appearance. In contrast to other MPM simulations handling fluids, we handle fluid and solid particles in the same grid pass. Moreover, in addition to tracking particle velocity and momentum, we extend FeatureSplatting's underlying Taichi-based MPM simulation [40] with an additional grid that tracks the mass of fluid being transferred between particles. These design choices lead to a reduction in complexity as opposed to multi-phase MPM simulations, without compromising on realistic and physics-based patterns of fluid transfer.

4.2.1. Simulation initialization

We use the information on the positions of all particles representing the scene obtained from 4.1 to initialize the particles in the MPM simulation. FeatureSplatting [9] transposes the Gaussian means into particle space by aligning all means to the ground plane, then uniformly scaling them to fit within a unit box, and finally translating them so that the scene is horizontally centered at (0.5,0.5) in the x-z plane and resting with its base at height 0.

All particles representing the scene are initialized as stationary, which means that their material is treated as such, and their velocity is zero. In addition, we initialize a horizontal grid of fluid particles parallel to the ground plane and at a pre-defined height that is configurable by the user. These fluid particles will represent the raindrops interacting with solid materials.

We extend the Taichi-based particle initialization process by tracking the porosity and saturation in the case of solid particles, and the fluid mass of all particles in the scene. Currently, all solid particles are initialized with the same porosity of 0.32. This value was inspired from [42] as the median porosity of gravel particles. The mass of fluid contained in all solid particles is assumed to be 0, and so is their saturation. The volume of solid particles, which limits the amount of fluid mass they can contain, is currently a configurable hyperparameter, measured in cm^3 and assumed to be consistent with real-world observations.

For raindrop particles, we initialize the saturation to 1. The fluid mass is computed as follows:

$$m_{\text{fluid}_i} = \rho_{\text{water}} \cdot \frac{4 \cdot \pi \cdot R_{\text{raindrop}}^3}{3} \quad (4.4)$$

Where ρ_{water} is assumed to be $1g/cm^3$, and $R_{\text{raindrop}} = 0.1025cm$. The choice for this value was empirical yet inspired by raindrop diameter measurements from [43].

4.2.2. Particle to Grid (P2G)

In our MPM simulation, solid particles transpose part of their fluid mass to the grid, but also gain solid mass from neighboring particles. On the other hand, since the simulation is not intended to model

puddle formation, fluid particles are assumed to diffuse entirely into solids they collide with. Due to this reason, the P2G step treats solid and fluid particles separately.

As per traditional MPM methods, the information transposed to the grid is dependent on the particle position. The method we are extending uses quadratic B-spline interpolation to map the fluid mass of a particle to the neighborhood of MPM grid nodes surrounding it. For a particle in a neighborhood of size 3, we add its fluid mass to the neighboring grid nodes similarly to accumulating the mass, as described in Section 3. This process is concretely formulated as:

$$fm_n = \sum_p w_n^{(p)} \cdot fm_p \quad (4.5)$$

Where fm_n represents the mass of fluid accumulated by grid node n , and $w_n^{(p)}$ represents the interpolation weight associated to particle p and grid node n .

The particle-to-grid transfer for fluid particles is largely similar, however we also enforce the constraint that fluid particles only scatter their mass to grid nodes containing solid particles. This ensures that the fluid mass of a raindrop is not scattered to other raindrops. To account for the cases in which interpolation weights do not sum to one, we normalize all weights when transferring fluid mass to the grid. Thus, Equation 4.5 is reformulated for fluid particles according to:

$$fm_n = \sum_p \frac{w_n^{(p)}}{\sum_n w_n^{(p)}} \cdot fm_p \quad (4.6)$$

Where notation remains largely the same, with the mention that grid node n is bound to contain solid particles.

4.2.3. Grid processing

Since we wish to capture realistic patterns of fluid dispersion among particles, we handle this implicitly via interpolation when transferring the fluid mass back to particle space. Transferring fluid directly between grid nodes will lead to block-like wetness patterns, which conflicts with reality. Thus, when processing the grid, we only ensure that the amount of fluid stored inside a node is not larger than the sum of total absorbed fluid mass per particle. We also assume that the total fluid mass is larger than 0. The capacity of particle p to hold fluid mass, namely the maximum value of fm_p , is dictated by Equation 2 from [15], namely:

$$fm_p \leq \rho_{water} \cdot \phi_p \cdot V_p \quad (4.7)$$

Where ϕ_p is the porosity of particle p , and V_p is its volume. Moreover, ρ_{water} maintains its value from subsection 4.2.1. Per the MPM initialization, we also currently assume all solid particles to have the same volume. The derivation of the final value at which the fluid mass of one grid node is capped can then be reformulated as:

$$fm_p \leq \rho_{water} \cdot \phi_p \cdot V \quad w_n^{(p)} \geq 0 \quad (4.8)$$

$$w_n^{(p)} \cdot fm_p \leq w_n^{(p)} \cdot \rho_{water} \cdot \phi_p \cdot V \quad (4.9)$$

$$\sum_p w_n^{(p)} \cdot fm_p \leq \sum_p w_n^{(p)} \cdot \rho_{water} \cdot \phi_p \cdot V \quad (4.10)$$

$$fm_n \leq \rho_{water} \cdot V \cdot \sum_p w_n^{(p)} \cdot \phi_p \quad (4.11)$$

Where fm_n represents the final capped value of grid node n , V is the volume assumed by all solid particles, and $w_n^{(p)}$ represents the interpolation weight associated to particle p and grid node n .

4.2.4. Grid to Particle (G2P)

In this simulation step, each solid particle gains fluid mass from its neighborhood on the MPM grid. We define a neighborhood of size 3, noted as $\mathcal{N}$, and again use quadratic B-spline interpolation with weights $w_n^{(p)}$ to transfer grid node values to the particle space. To avoid an undesirable increase of fluid mass values among particles, we subtract the mass that has been deposited to the grid, $fm_{scatter}$, from the final result. The final fluid mass obtained per particle, $\hat{m}_p$, is expressed as:

$$\hat{m}_p = \sum_{n \in \mathcal{N}} w_n^{(p)} \cdot fm_n - fm_{scatter} \quad (4.12)$$

$$0 \leq \hat{m}_p \leq \rho_{water} \cdot \phi_p \cdot V \quad (4.13)$$

Where $\mathcal{N}$ represents the neighborhood of size 3 surrounding the grid node with fluid mass fm_n , associated with particle p , that has porosity ϕ_p and volume V .

Fluid particles are assumed to disperse their entire mass, fm_p , into solid materials. Thus, they do not accumulate any mass in the G2P step, rather they only lose the mass that has been scattered to the grid in the P2G step, namely $fm_{scatter}$:

$$\hat{m}_p = fm_p - fm_{scatter} \quad (4.14)$$

$$fm_p \geq 0 \quad (4.15)$$

In the G2P step of the simulation, the saturation of each solid particle is computed as a function of its final fluid mass. This value will subsequently inform the appearance of 3D Gaussians representing the scene. We take inspiration from [15] and compute the final saturation of particles as follows:

$$s_p = \frac{fm_p}{\rho_{water} \cdot \phi_p \cdot V} \quad (4.16)$$

$$0 \leq s_p \leq 1 \quad (4.17)$$

Where s_p represents the final saturation value of a solid particle with fluid mass fm_p , porosity ϕ_p and volume V at the end of an MPM simulation step.

4.3. Appearance parameter update

The postprocessing step in the pipeline leverages the per-Gaussian saturation values in order to alter their appearance in a physically-grounded manner. Wet materials can be thought of as covered by a thin film of water. This increases the specularity of the material on the one hand, and renders it darker on the other hand. The darkening effect occurs due to the repeating light reflectance between the material and the water-air interface, allowing the surface to absorb more light [44]. Ideally, the albedo and specularity of 3D Gaussians would be easily accessible and modifiable. However, this information is not intuitively encoded in the color representation based on Spherical Harmonics stored within the Gaussians in the conventional formulation of Gaussian splatting [6]. To directly modify a Gaussian's albedo and specularity, more complex representations such as Bidirectional Reflectance Distribution Functions (BRDFs) would be required. As we lack such a representation, we only modify the Gaussian color and opacity.

We approximate the per-Gaussian 0-level SH coefficients as the albedo, and we operate under the assumption that all materials in the scene are fully dry prior to contact with the rain particles. Following [45], we model the fully saturated appearance of a Gaussian ($\bar{w}_{0wet}$) as a function of its dry albedo ($\bar{w}_{0dry}$) based on the ratio r of the particle refractive index to that of the surrounding medium:

$$\bar{w}_{0wet} = \frac{r\bar{w}_{0dry}}{r\bar{w}_{0dry} + 1 - \bar{w}_{0dry}} \quad (4.18)$$

$$r = \frac{1 - g_{wet}}{1 - g_{dry}} \quad (4.19)$$

Where g_{wet} and g_{dry} are the average cosines of the scattering angles of the wet and dry materials, respectively. In the base implementation, we assume a single material and, as such, these indices are hardcoded.

To account for the per-Gaussian saturation levels, we model the material appearance every time step in terms of the computed dry and wet albedos. The final value that overrides the per-Gaussian spherical harmonics ($\hat{s}h_{(0,g)}$) is computed as:

$$\hat{s}h_{(0,g)} = \bar{w}_{0dry} + s_g \cdot \bar{w}_{0dry} \quad (4.20)$$

In order to accentuate the darkness of wet materials, we also increase the opacities of saturated Gaussians. This value increases inversely proportional to the saturation level in the particle associated to each Gaussian.

5

Experimental Setup and Results

In this section, we evaluate our approach under different experiments. We begin by outlining the hypotheses we aim to test, then describe the scenes input to the simulation, and follow with the setup of each experiment, explaining how each test contributes to validating the approach.

As the main contribution of this study is the augmentation of the MPM simulation (see Section 4.2), the experimental setup will primarily focus on its validation. The final goal of the study is to simulate a fluid flow through porous media that is realistic and, to an extent, physically accurate. Consequently, we expect that fluid spreads radially outwards, from the point of impact to its neighbors, upon the raindrop making contact with the solid surface. We also expect particles closer to the point of impact to saturate before particles further away do. Different materials should also produce different patterns of fluid spread. Lastly, we aim to test the impact of the density of the input point cloud and of the angle of impact. With these experiments, we aim to answer RQ3 and RQ4 (see Chapter 1).

5.1. Input data and preprocessing

In our experiments, we prioritize testing the presented method under both a 2D and a 3D setup. To verify whether our approach correctly handles ideal data and to enable a higher level of control over the experimental setup, we chose to create synthetic scenes with varying levels of geometric complexity. It is important to note that, following the output of the preprocessing pipeline, the positions of points across all input scenes are constrained between 0.0 and 1.0 across the X, Y, Z axes. Due to missing knowledge on initial scene measurements, we cannot infer concrete dimensions across the input point clouds in terms of a unit of measurement.

Below we provide a description of each scene, alongside any necessary preprocessing, a few training images used to train the Gaussian Splats, and renders obtained through FeatureSplatting’s 3DGS backbone, namely Splatfacto [46].

5.1.1. Horizontal 2D Grid Scene

The horizontal grid is a toy input meant to validate basic physical constraints that should be fulfilled by the simulation, as well as study the effect of varying multiple parameters on the final patterns of fluid spread. The scene includes a square representing the ground plane, with a side length of 0.4. The initial positions of all particles have X and Z coordinates in the 0.3 - 0.7 range, and a Y-value of 0.0. We add one raindrop at a height of 0.2 on the Y-axis and set it to fall towards the ground plane with a speed of 9.8 m/s. For instance, given a toy grid containing x particles across one side, the point cloud representing the grid will have x^2 particles. A top view of the initial state of the simulation is provided in Figure 5.1.

With this setup, we aim to study the geometric pattern of fluid spread from the point of impact to neighboring particles. We expect particles to be saturated after contact with water, and the fluid to spread outward from the point of impact in concentric circles. We also aim to test the impact of multiple parameters on the simulation results, namely: the resolution of the MPM grid, the density of the toy grid, the

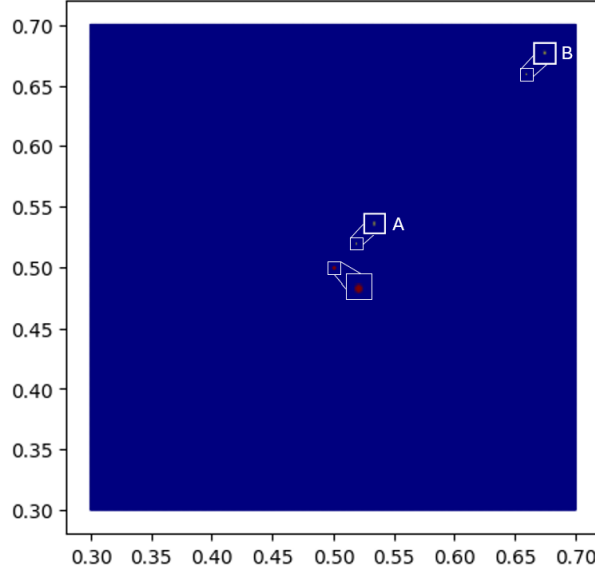


Figure 5.1: Capture of the initial state of the simulation. Highlighted in red is the raindrop. Highlighted in white are points A and B, which will track evolutions in particle saturations at different positions along the grid. This grid contains 400x400 particles.

porosity of the solid particles, and the angle of impact between the raindrop and the solid surface.

5.1.2. Synthetic Sponge Scenes

This scene visualizes a cubic sponge on flat wooden flooring, surrounded by white walls. Its goal is to evaluate the preprocessing pipeline, as well as validate hypotheses related to the fluid flow, in a simple three-dimensional environment. We provide renders of the scenes generated in Blender [47], alongside renders of the initial 3D Gaussian splat generated in NeRFStudio [48] (see Figure 5.2).



Figure 5.2: Renders of the synthetic cube scene. On the left: two training scenes obtained from Blender [47]. On the right: their equivalent renders obtained by 3D Gaussian Splatting using Splatfacto [46].

Preprocessing Pipeline

To substantiate the design choices made for this pipeline component, we run a series of experiments focusing specifically on scene preprocessing. We take as input a pre-trained 3D Gaussian representation of the synthetic cube scene and capture the resulting point clouds. We conduct an ablation study which toggles off different preprocessing components and compares the obtained results. Thus, the contribution of each pruning and infilling strategy highlighted in Section 4.1 can be clearly visualized.

To verify the pruning pipeline, we run the process by toggling each of its components in turn. This

includes pruning the Gaussians based on transparency and size, then filtering them based on semantic similarity and spatial distance. The results of this process are highlighted in Figure 5.3.

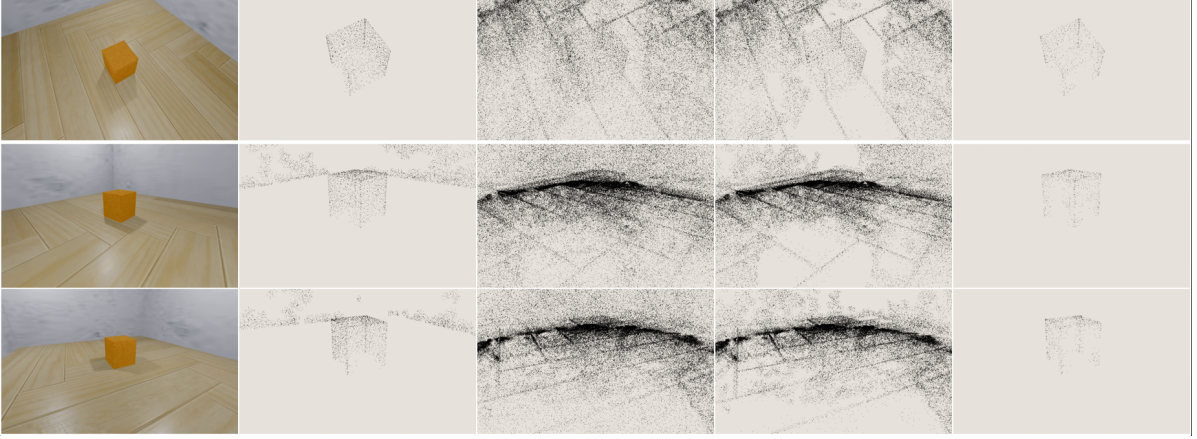


Figure 5.3: Results of an ablation study highlighting the contributions of multiple pruning strategies to the final result. From left to right: (1) splats of the Gaussians taken as input, point clouds obtained when bypassing the following components in the preprocessing pipeline: (2) pruning by transparency and size, (3) outlier removal in terms of spatial distance, (4) outlier removal in terms of semantic similarity, and (5) final result with all pruning strategies combined.

We proceed by focusing on the infilling component. Figure 5.4 shows the final results of infilling applied to the cube and, separately, to the ground plane. Moreover, Figure 5.5 presents the results of an ablation study run on the ground plane of the synthetic cube scene, comparing different interpolation methods for approximating the flat surface (refer to Section 4.1 for implementation details).

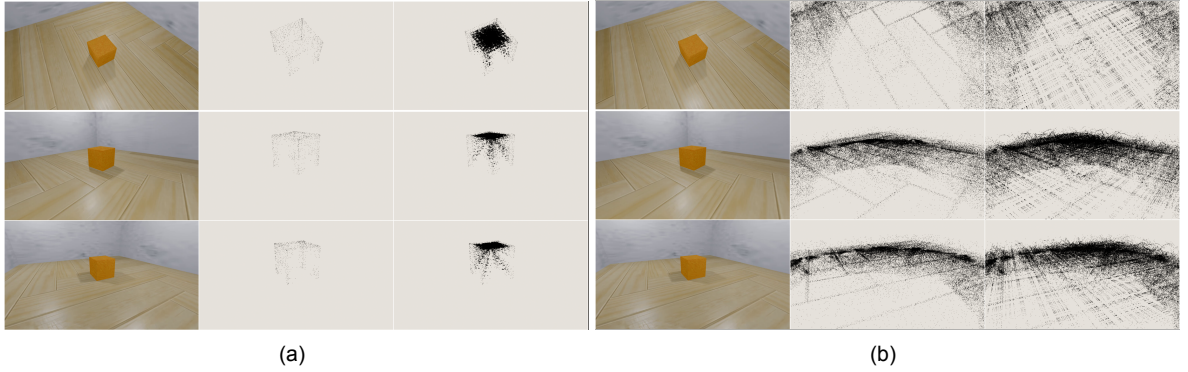


Figure 5.4: Result of infilling pipeline for (a) above-ground objects and (b) the ground plane in the synthetic sponge scene. From left to right in (a): (1) splats of Gaussians taken as input, (2) final result of pruning pipeline, (3) final result of object infilling pipeline. From left to right in (b): (1) splats of Gaussians taken as input, (2) point cloud capture of the ground before infilling, (3) final result of object infilling pipeline.

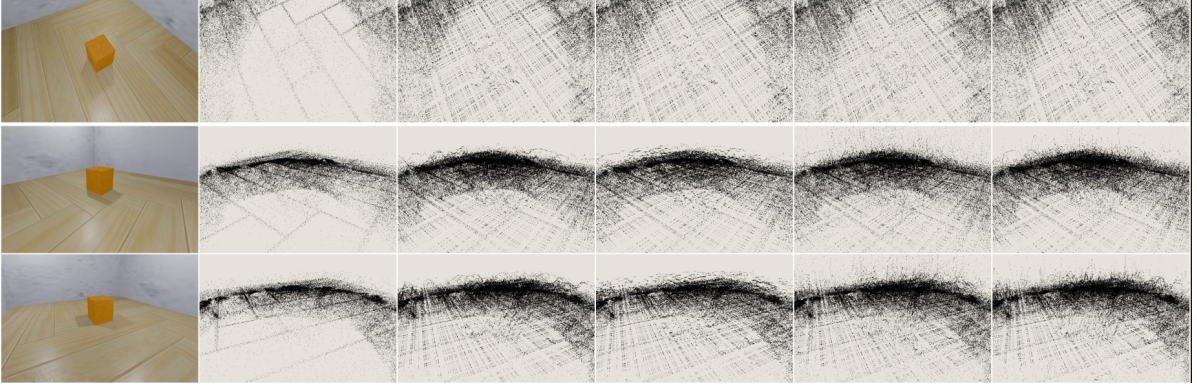


Figure 5.5: Result of using different interpolators when infilling the ground particles in the synthetic sponge scene. From left to right: (1) splats of Gaussians taken as input, (2) point cloud capture of the ground before infilling, and infilling results when using a: (3) Linear interpolator, (4) Nearest Neighbor interpolator, (5) RBF interpolator using a Gaussian kernel, (6) Clough-Tocher interpolator.

Ablation: Capture of Different Materials

In order to test the capability of the preprocessing pipeline to capture multiple materials, we modify the synthetic cube scene highlighted in Figure 5.2. This variant contains four cubes of different materials: wood, claystone, plastic and concrete. The texture of the floor is also modified to simulate soil. Figure 5.6 visualizes the original captures alongside the obtained Gaussian Splats.

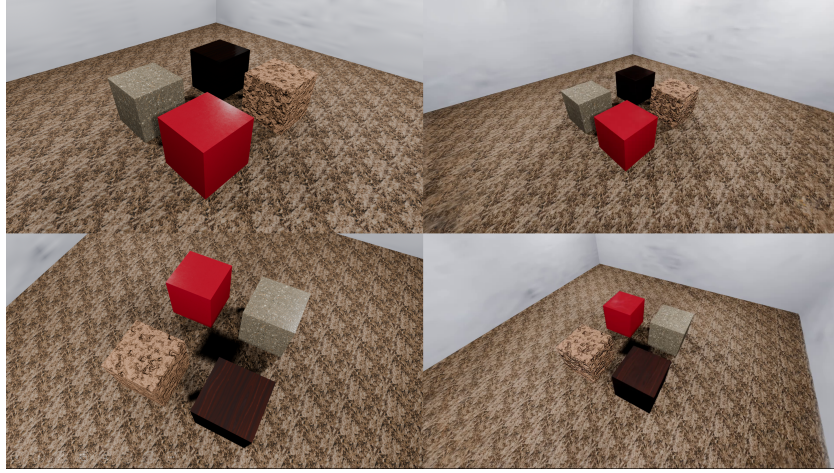


Figure 5.6: Renders of a synthetic cube scene containing multiple materials. On the left: two training scenes obtained from Blender [47] On the right: their equivalent renders obtained by 3D Gaussian Splatting using Splatfacto [46].

We extend the preprocessing pipeline by associating a material with each particle that has been selected for simulation. For particles derived from 3D Gaussian positions, we use FeatureSplatting’s [9] pre-derived distilled features per 3D Gaussian to compare the semantic similarity between each Gaussian and a list of pre-defined materials. Following FeatureSplatting’s [9] approach, we embed all material names (see Table 5.1) using a CLIP text encoder, then compute pairwise cosine similarities between the Gaussian distilled features and the text embeddings. The similarities are then transformed into a probability distribution using the softmax function, then each Gaussian is associated to the material with the highest similarity value.

We initialize the material of support particles added in the infilling step as that of the majority of the Gaussians associated with the infilled object. Each material is given a specific porosity ϕ . All values are derived from existing literature; Table 5.1 presents a list of all of the materials considered for this ablation.

We apply this extended preprocessing pipeline to the modified synthetic cube scene highlighted in Figure 5.6. Results of this experiment are highlighted in Figure 5.7.

Material	ϕ
Soil	0.06 [49]
Softwood	0.70 [50]
Claystone	0.11 [51]
High Strength Design Concrete	0.03 [52]
Plastic (Polypropylene)	0.13 [53]

Table 5.1: Porosity values of all materials considered in the ablation study.

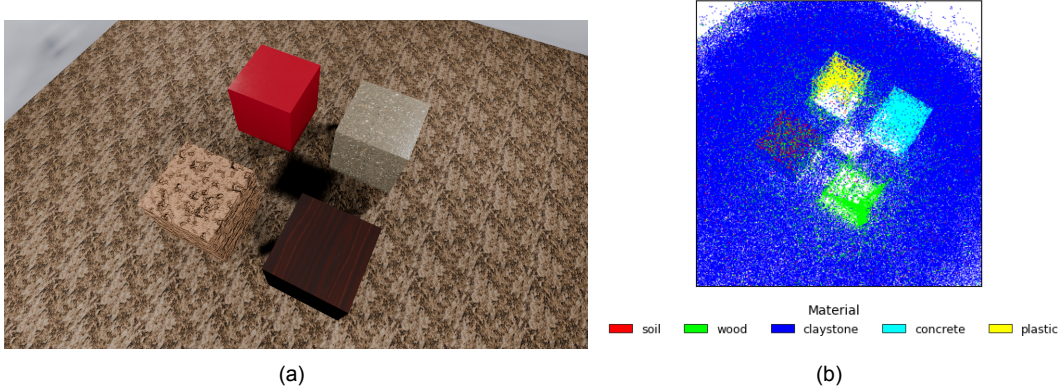


Figure 5.7: Result of the ablation identifying multiple materials from the modified synthetic cube scene. On the left: original training scene for the 3D Gaussian Splat. On the right: Point cloud obtained from preprocessing the Gaussian Splat, applying infilling per object, and categorizing each particle into a different material.

5.2. MPM simulation

To test the MPM simulation, we devised multiple scenarios increasing in complexity, which aim to test the influence of different parameters pertaining to the simulation and the input scene on the final result. We specifically focus on the resolution of the MPM grid and the density of the input point clouds. These factors most directly influence the results of the simulation. We provide graphs of the increase in mean saturation, as well as heatmaps showing clear patterns of evolution in the saturation values over time. We also provide graphs showing the progression of saturation values in two arbitrarily chosen points within the input point cloud. We report these results per scenario.

5.2.1. Horizontal 2D Grid

We begin by running a ten-step simulation of the input setup described in 5.1. For all experiments involving homogeneous materials, we initialize all solid particles with the same porosity of 0.32. This value was inspired from [42] as the median porosity of gravel particles.

In our first experiment, we vary the resolution of the MPM grid, as well as the density of the given input grid, and report captures of the simulation state at each of the 10 time steps. For this experiment, we let one raindrop fall towards the center of the 2D grid, and observe that the impact occurs between step 4 and step 5. More concrete results are shown in Figure 5.8.

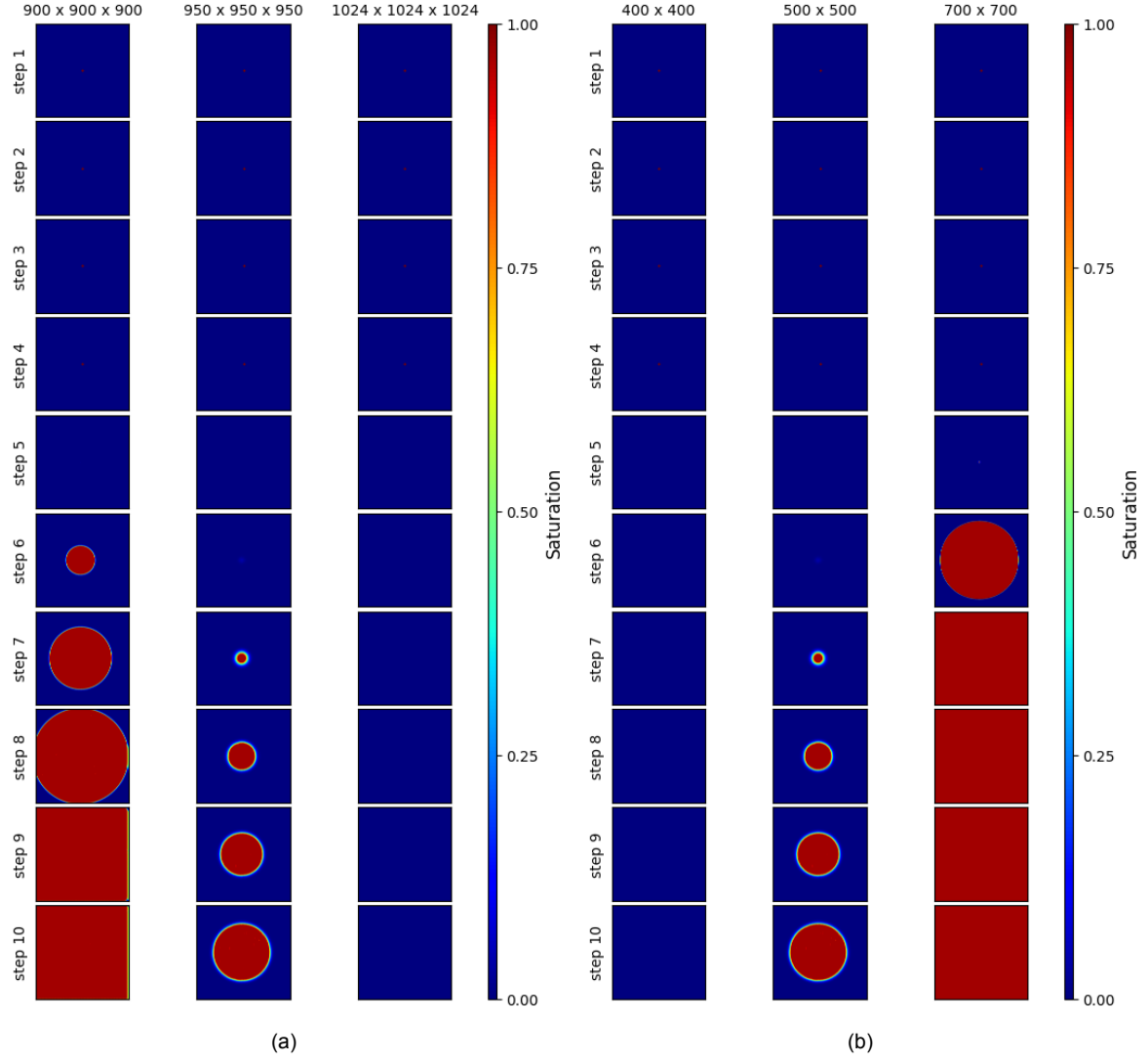


Figure 5.8: Top-side captures of point clouds showing the simulation state throughout the ten steps. The heatmaps on the right of each figure correlate the saturation values of each particle in the simulation to a specific color. In Figure (a), we vary the MPM grid resolution on an input of 500x500 particles. In Figure (b), we vary the number of particles in the input point cloud (400x400, 500x500 and 700x700 particles), using an MPM grid resolution of 950x950x950.

To further verify that the fluid spreads outward from the raindrop's point of contact with the flat surface, we measure the saturation values at points A and B (per Section 5.1) throughout the ten-step simulation. Figure 5.9 showcases the increase of saturation depending on the MPM grid resolution and input density.

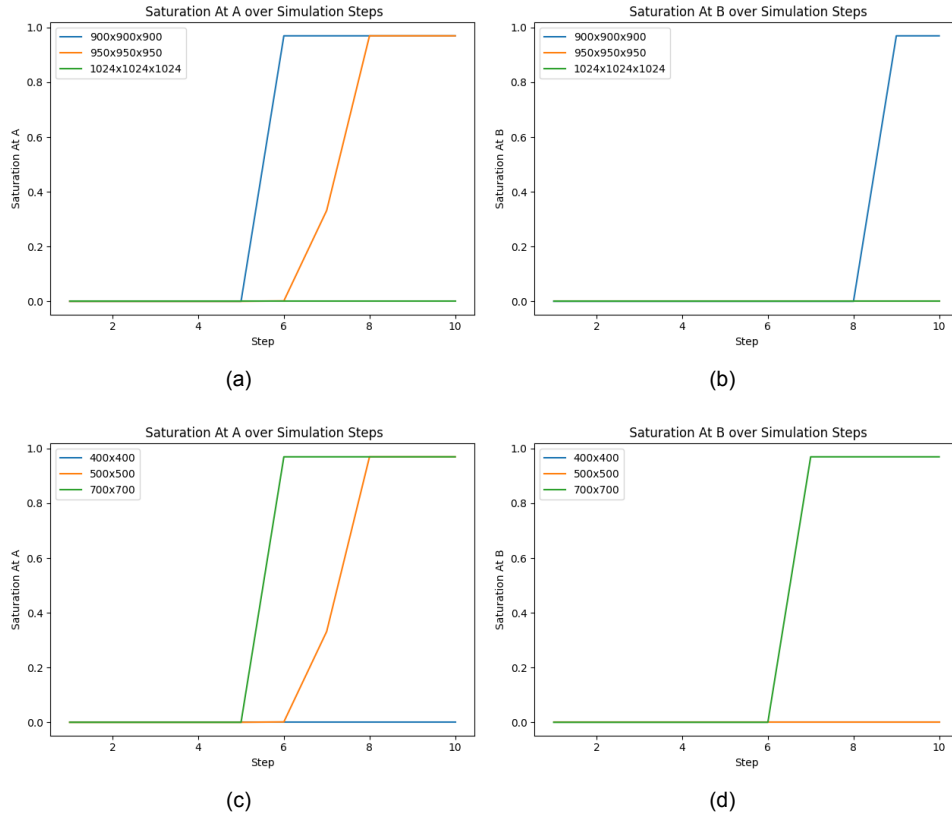


Figure 5.9: Plots of the saturation measured at points A and B within the input grid (see Figure 5.1). We compare varying the MPM grid resolution in (a) and (b), and varying the number of particles in the input point cloud in (c) and (d).

To study the influence of the angle of impact on simulation results, we tilt the input grid by 20% in the x-direction and re-run the same simulation. Figure 5.10 highlights point cloud captures when comparing MPM grid resolutions and input grid densities, similarly as before.

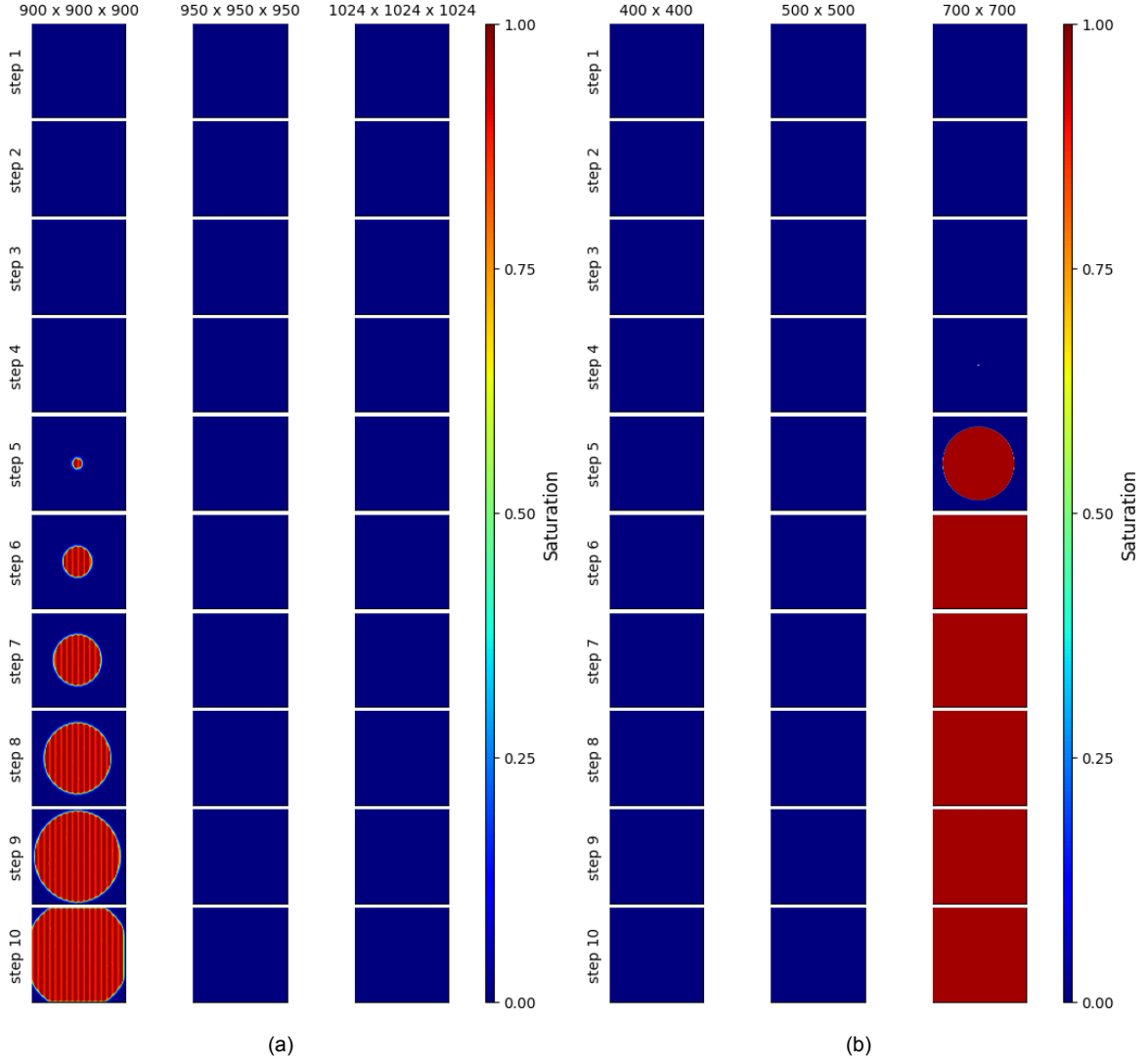


Figure 5.10: Top-side captures of point clouds showing the simulation state throughout the ten steps. The heatmaps on the right of each figure correlate the saturation values of each particle in the simulation to a specific color. In Figure (a), we vary the MPM grid resolution on an input of 500x500 particles. In Figure (b), we vary the number of particles in the input point cloud (400x400, 500x500 and 700x700 particles), using an MPM grid resolution of 950x950x950.

In order to capture the interaction between patterns of fluid transfer when multiple raindrops are involved, we place four raindrops at a distance of 0.1 from the horizontal grid, then run the simulation for ten steps. The final result is highlighted in Figure 5.11.

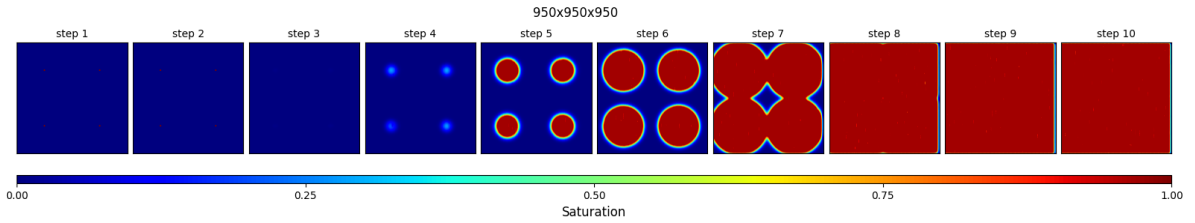


Figure 5.11: Top-side captures of point clouds showing the simulation state throughout the ten steps. The heatmap below correlates the saturation values of each particle in the simulation to a specific color. This simulation was run on a point cloud containing 500x500 particles, with an MPM grid resolution of 950x950x950.

Moreover, to confirm whether different materials behave differently, we take a 2D square grid and

initialize its left half with the porosity of softwood, and the right half with the porosity of soil, as per Table 5.1. Results of this experiment are reported in Figure 5.12.

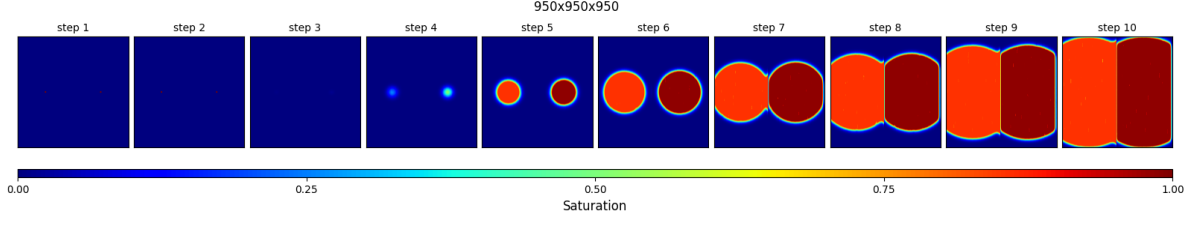


Figure 5.12: Top-side captures of point clouds showing the simulation state throughout the ten steps. The heatmap below correlates the saturation values of each particle in the simulation to a specific color. This simulation was run on a grid of 500x500 particles, with an MPM grid resolution of 950x950x950.

Lastly, we provide a comparison of times taken per simulation step, given the horizontal grid, in Table 5.2. We compare the impact of different resolutions for the MPM grid in terms of number of grid nodes per dimension over a horizontal grid of 500x500 particles. We also compare running a simulation on an MPM grid containing 950x950x950 nodes, on a horizontal grid defined by 400x400, 500x500 and 700x700 particles respectively.

MPM Grid resolution	Avg. time per step (s)	Input grid density	Avg. time per step (s)
900x900x900	2.06	400x400	2.05
950x950x950	2.25	500x500	2.25
1024x1024x1024	2.54	700x700	2.62

Table 5.2: Comparison of time in seconds taken per MPM simulation step. On the left: comparison between different MPM grid resolutions. On the right: comparison between different densities in the input 2D Grid.

5.2.2. Synthetic cube

In this scenario, we run the pre-processing pipeline on the 3D Gaussian Splat of the cube scene (see Figure 5.2), then feed its particle representation into the MPM simulation. Similarly as before, we simulate one raindrop falling onto the center of the scene at a speed of 9.8 m/s. To study the impact of the input density similarly as in the 2D case, we introduce support particles additional to the ones added in the preprocessing pipeline, per Section 4.1. Within the bounding box of the cube, we introduce a 3D grid of particles with varying density. Similarly to the horizontal grid setup, we initialize the porosity of all solid particles in homogeneous materials as 0.32. Figure 5.13 shows capture of the simulation input, alongside the number of particles introduces per one side of the cube.

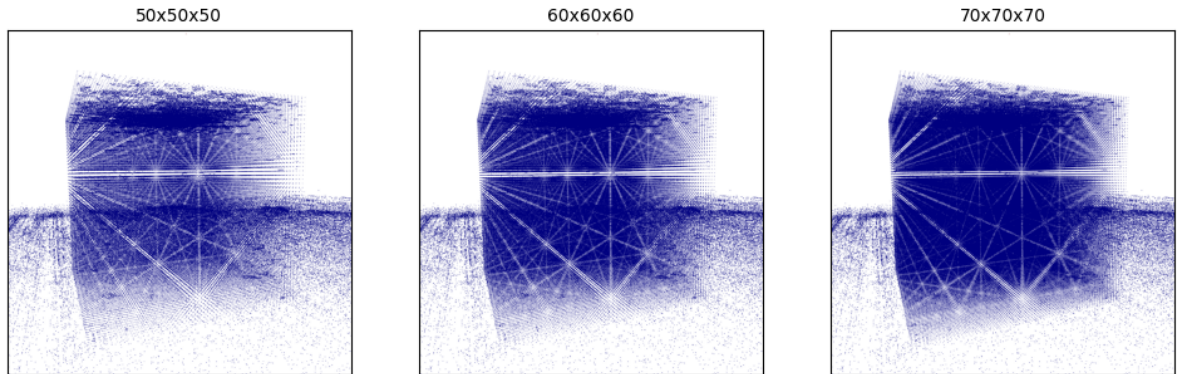


Figure 5.13: Capture of the initial state of the simulation. From left to right: infilling the synthetic cube with a number of 50x50x50, 60x60x60 and 70x70x70 particles respectively.

We vary the resolution of the MPM grid and report the results of running the simulation for ten steps. In addition, we run the ablation that fills in the cube with a 3D grid consisting of 50x50x50, 60x60x60 and

70x70x70 particles respectively. We report a comparison of different MPM grid sizes in Figure 5.14, as well as different cube densities in Figure 5.15.

Finally, to validate the capability of the simulation to handle diverse materials, we reproduce the point cloud illustrated in Figure 5.7 (b) and input it to a ten-step simulation. The focus of this experiment is to verify the behaviour of the MPM simulation with respect to the material-aware ablation described in Subsection 5.1.2, under the assumption that the preprocessing pipeline produces an ideal output. We construct the ground plane as a horizontal grid of 300x300 particles with XZ coordinates spanning the range [0.0, 1.0]. Four cubic volumes are then placed onto the ground plane, centred at coordinates (0.4, 0.4), (0.4, 0.6), (0.6, 0.4), and (0.6, 0.6), each with a side length of 0.1 and represented by sets of 30x30x30 particles. Lastly, we position four rain particles above the centre of each cube at a vertical distance of 0.1, with an initial downward velocity of 9.8 m/s. The simulation was run on an MPM grid of resolution 600x600x600.

We visualize the initial state of the simulation is visualised in Figure 5.16, and report the results of this experiment in Figure 5.17.

5.3. Appearance parameter update

In this section, we provide captures of the final state of the physics simulation, including the appearance parameter update, as rendered in NeRFStudio [48]. The full pipeline is evaluated on the synthetic cube scene (see Figure 5.2), to which we add 900 raindrops. The raindrop Gaussians are initialized in a horizontal grid at a distance of 1.0 above the ground, and assigned an initial downward velocity of 9.8 m/s. Taking inspiration from Let it Snow! [11], we set the XYZ scales of the Gaussians to [0.002, 0.002, 0.006], the Gaussian opacity to 0.25 and the color to [226.75, 221.85, 229.5]. Lastly, per Section 4.3, we empirically chose a value of 0.5 for g_{dry} and a value of 0.97 for g_{wet} .

We capture the last scene before impact, then the first four scenes after the first raindrops make contact with the scene. Results of this experiment are shown in Figure 5.18.

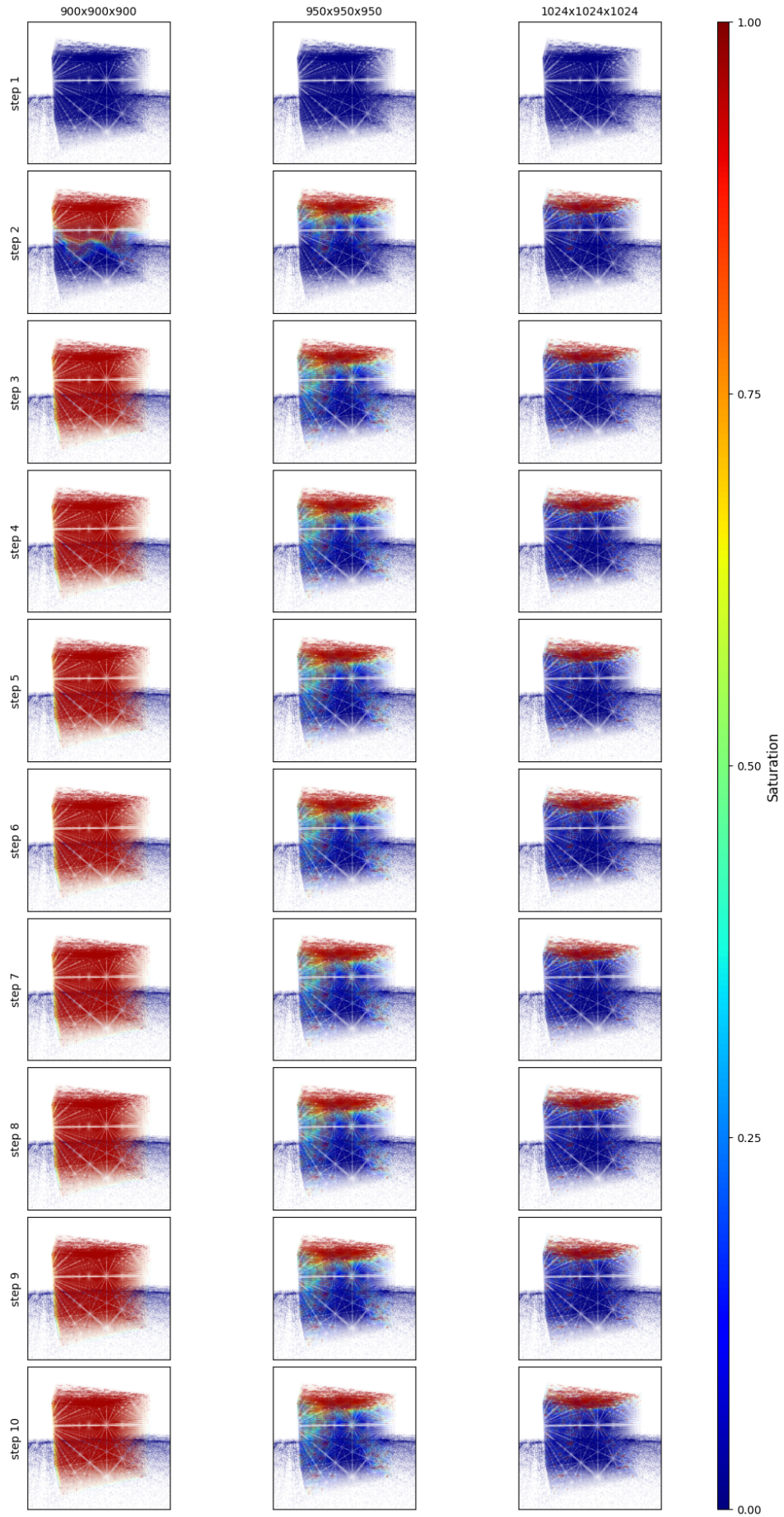


Figure 5.14: Lateral captures of point clouds showing the simulation state throughout the ten steps. The heatmap on the right correlates the saturation values of each particle in the simulation to a specific color. In this experiment, we vary the resolution of the MPM grid, between 900x900x900, 950x950x950 and 1024x1024x1024 grid nodes. The three simulations were run on the preprocessed cube scene with a grid-based infilling consisting of 60x60x60 particles.

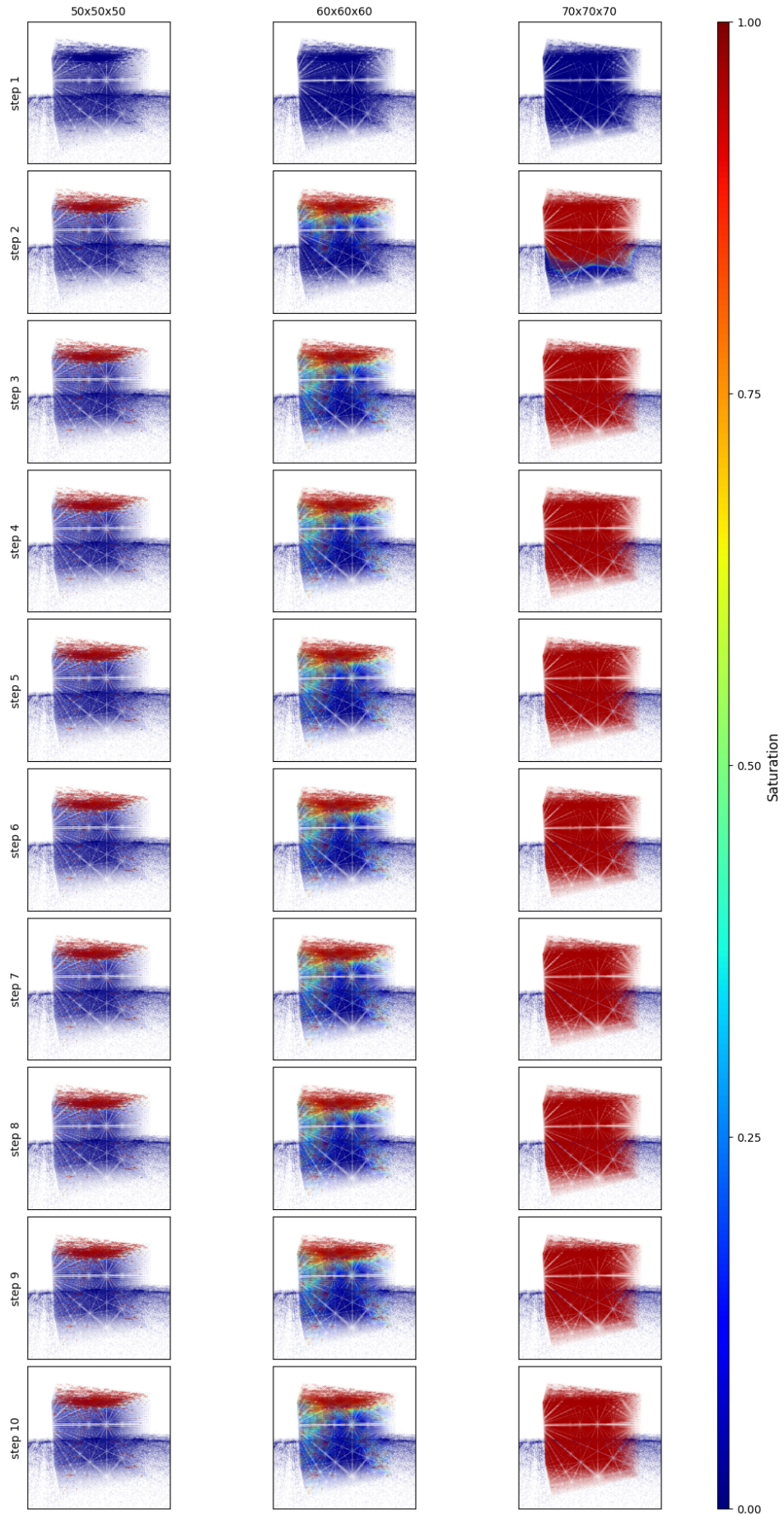


Figure 5.15: Lateral captures of point clouds showing the simulation state throughout the ten steps. The heatmap on the right correlates the saturation values of each particle in the simulation to a specific color. In this experiment, we vary the density of grid-based infilling applied on the cube scene, between 50x50x50, 60x60x60, and 70x70x70 particles. The simulations highlighted in this figure were run on an MPM solver with a grid resolution of 950x950x950.

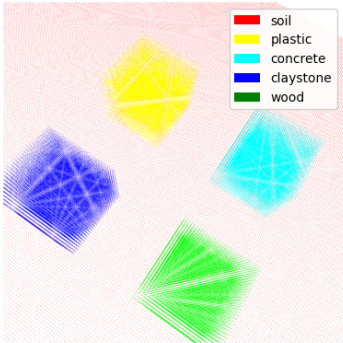


Figure 5.16: Capture of the initial state of the simulation testing the material-aware ablation in 3D. The legend highlights the material that each color corresponds to.

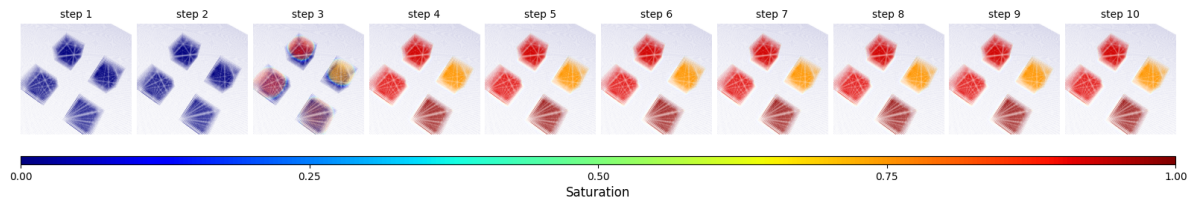


Figure 5.17: Top-side captures of point clouds showing the simulation state throughout the ten steps. The heatmap below correlates the saturation values of each particle in the simulation to a specific color. We run this experiment on an MPM grid of resolution 600x600x600.

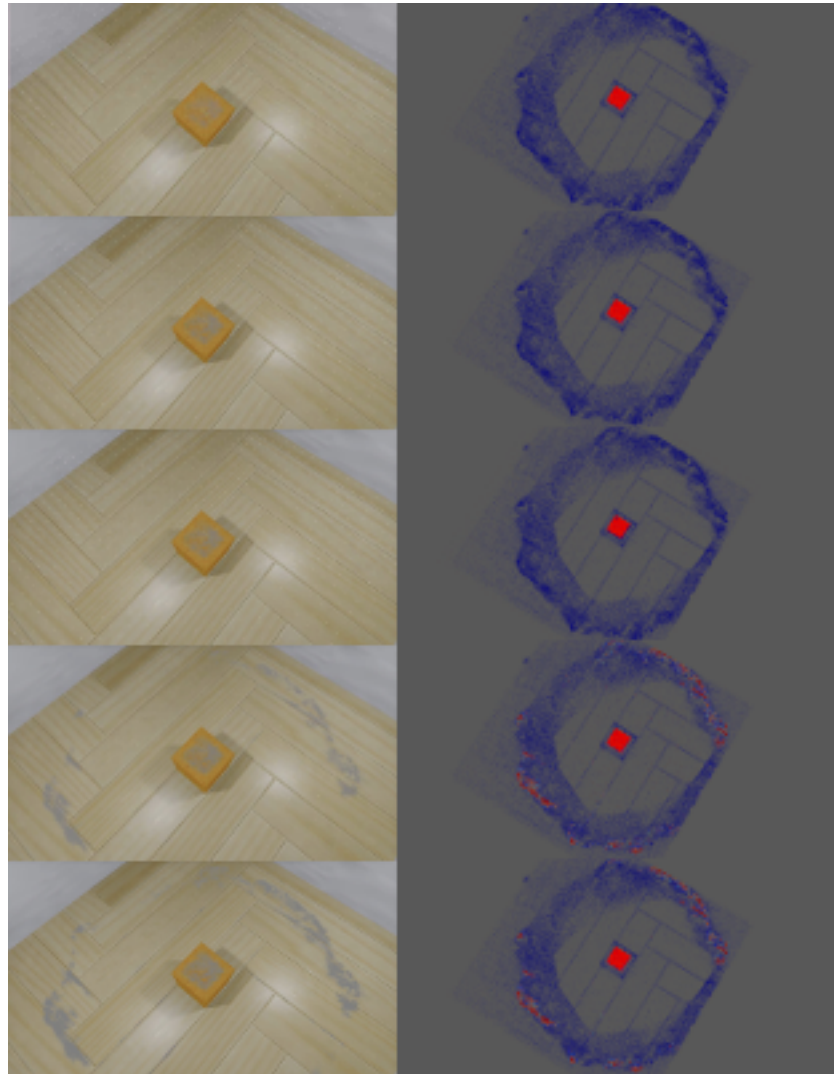


Figure 5.18: Result of the end-to-end pipeline, based on a Gaussian Splat of the synthetic cube scene (See Figure 5.2). On the left: captures of the first five simulation steps following the initial impact of the raindrops with the scene. On the right: captures of the point clouds highlighting the current simulation state

6

Discussion

In this section, we draw a few conclusions based on the characteristics, strengths, weaknesses and limitations of our methodology, given the MPM simulation results highlighted in Section 5.2. The proposed MPM simulation transfers fluid mass from one particle to its spatial neighbors by interpolation. The value of the fluid mass contained by a particle informs its final saturation, and is constrained to be less than or equal to its maximum capacity, which is determined by the particle’s porosity and volume. The goal of the proposed methodology was to create a simulation that is compatible with a Gaussian Splatting pipeline; taking this into account, complex variants of the MPM such as multi-point MPMs were considered out of scope. However, this produces some limitations; for instance, the proposed methodology does not simulate phenomena such as puddle accumulation.

We aimed to test the method in a few simple setups which clearly highlight the impact of specific hyperparameters and types of input data. Each experiment yielded different findings pertaining to the initial hypotheses that were made, allowing us to answer RQ3 and RQ4 (see Chapter 1). We extract the main learnings in Section 6.1, then focus on the limitations in Section 6.2.

6.1. Discussion of results

We begin with a brief discussion of the preprocessing pipeline. As evidenced by Figure 5.3, each pipeline component contributes meaningfully to the overall result. Most notably, bypassing pruning by spatial distance or pruning by semantic similarity results in the most artifacts. Pruning by transparency and size appears to have the least impact on the pipeline output; however, it remains a valuable component, as it substantially reduces the number of Gaussians that must subsequently be filtered by DBSCAN. Nevertheless, the pipeline can at times be overly stringent, resulting in the erroneous exclusion of Gaussians belonging to the target object. Additionally, given the planar geometry of the ground surface, certain ground Gaussians may be inadvertently removed during the transparency and size pruning stage, yielding a less accurate representation of the ground plane.

Regarding the infilling component (see Figures 5.4 and 5.5), a strategy approximating simple planar surfaces was implemented to accommodate the synthetic cube scene used for evaluation purposes. The original infilling pipeline is better suited to high-frequency geometries, such as grass or vegetation, and exhibits reduced performance on planar surfaces. However, since fluid transfer is measured upon impact with the top surface of the cube, this infilling strategy is considered sufficient for the purposes of this evaluation. With respect to the ground plane, the infilling component proves particularly effective in filling the gaps introduced by the pruning stage. Ultimately, the quality of the simulation input is inherently dependent on the underlying 3D Gaussian Splatting representation.

Based on the results in Figures 5.8 and 5.10, the proposed MPM model treats fluid flow as radial, extending outward from the point of contact at a constant rate. We also notice a halo surrounding the pattern of saturation, yet its size seems to also depend on the rate at which fluid spreads. This result is expected for a homogeneous material as it demonstrates the absorption of fluid mass into the solid medium, as well as the outward expansion in the wetness pattern. The shape of the patterns of fluid

flow is determined by the type of interpolation used to transfer fluid between particles, namely quadratic B-spline interpolation. The usage of an interpolation function with a continuous derivative minimizes grid-crossing errors, leading to a smoother result [54], as demonstrated in the heatmaps highlighted in Subsection 5.2.1.

With regard to the horizontal grid, we also notice that particles closer to the point of contact saturate faster than those further away. This is confirmed on the one hand by the heat maps capturing particle saturations throughout the simulations involving the horizontal grid, but also when we track saturations at specific points in the input point cloud. Figure 5.9 plots the saturation values at point A, closer to the point of impact, and point B, which is further away. Upon studying the plots, we notice that, in the cases in which particles do saturate, point A will always saturate faster than point B. This result is expected and it concretely demonstrates the outward expansion of the wetness pattern.

The experiments carried out on the horizontal grid also distinctly highlight the main factors influencing the physics simulation results, providing necessary insight to answer RQ4 (see Chapter 1). Figures 5.8 and 5.10 bring to light two main findings in this regard. On the one hand, varying the resolution of the MPM grid has a direct impact on the rate of the fluid spread. The size of the MPM grid resolution is inversely proportional to the rate at which fluid spreads within the solid material, as noticeable in Figure 5.8 (a). The result is intuitive; with a denser MPM grid, node neighborhoods span a smaller distance. Since one particle spreads fluid to a neighborhood of nodes of fixed size, fluid will incrementally spread over smaller distances. The denser the grid, the smaller the rate at which the fluid spreads; this is especially apparent when increasing the MPM grid resolution to $1024 \times 1024 \times 1024$ nodes, where we notice that the material does not saturate.

On the other hand, Figure 5.8 (b) reveals that the density of the input point cloud influences the rate of fluid spread to an even larger extent than the number of MPM grid nodes. This behavior is also expected. With a sparser point cloud, MPM grid nodes will accumulate less information on the quantity of fluid in particles within their neighborhood. This reduces the rate of fluid spread, leading to no visible saturation patterns when the input point cloud becomes too sparse.

Figure 5.10 illustrates the effect of varying the angle of impact between the raindrop and the solid material. As observed in Figure (a), tilting the horizontal grid introduces several artifacts in the fluid transfer patterns, manifesting as alternating lines of elevated and reduced saturation across the spread region. These artifacts, absent in the horizontal grid results presented in Figure 5.8, can be attributed to the misalignment between the input point cloud orientation and the horizontal layout of the MPM grid, which produces irregular interpolation patterns. The overall trends in fluid spread rate remain consistent with those observed in the horizontal grid experiment, with the notable exception that the effects of increasing the MPM grid resolution and input point cloud density on the rate of fluid spread appear to be more pronounced in this configuration.

When introducing multiple raindrops in a simulation homogeneous materials, as per Figure 5.11, we notice that there is no difference between fluid spread patterns emerging from the points of contact with the solid surface. However, saturation patterns emerging from multiple points seem to merge into each other. We consider this process realistic and explainable by the smoothing resulting from B-spline interpolation.

In addition, Figure 5.12 provides evidence on the behavior of the simulation under heterogeneous material conditions. We notice that the spatial spread of fluid is not affected on materials of different porosities, however the recorded saturations vary accordingly. This is consistent with our expectations; following Equation 4.16, the final saturation value is inversely proportional to the material porosity. Consequently, given an equal fluid mass, a material of higher porosity will record a lower saturation value, which aligns with the physical intuition that a particle with greater porosity possesses a higher capacity for holding water and therefore saturates at a slower rate. Another observation drawn from Figure 5.12 is that patterns of fluid flow do not merge similarly to Figure 5.11, nor does fluid mass appear to redistribute from saturated particles to those of lower saturation. A plausible explanation for this phenomenon lies in the upper bound imposed on the fluid mass a single grid node may accumulate during the grid processing stage. This behavior is acknowledged as an inherent tradeoff of the current formulation.

When extending the simulation to a three-dimensional environment, additional insight into the fluid

transfer behavior is obtained. We find the extent of fluid mass propagation to be contingent on the local particle density in the neighborhood of saturated particles. From Figures 5.14 and 5.15, it is concluded that the majority of simulations converge by step 2, with only marginal changes observed in subsequent steps. This rapid convergence is attributed to the spatial proximity of neighboring particles. Particles in close proximity to one another saturate quickly, whereas those located at greater distances receive fluid from fewer neighbors, resulting in slower saturation. This behavior is further substantiated by the spatial pattern of fluid flow, which converges around particles introduced during the original infilling pipeline (see Figure 5.4), as these particles reside in regions of higher local density.

Furthermore, Figure 5.15 reveals pronounced differences in simulation results across the three levels of grid-based infilling evaluated ($50 \times 50 \times 50$, $60 \times 60 \times 60$, and $70 \times 70 \times 70$ particles). These discrepancies are attributed to the substantial variation in total particle count between the respective infilling levels. Specifically, the transition from $50 \times 50 \times 50$ to $60 \times 60 \times 60$ introduces an additional 91,000 particles, while the transition from $60 \times 60 \times 60$ to $70 \times 70 \times 70$ results in an increase of 127,000 particles. It is therefore concluded that these large inter-level differences in particle count are the primary cause of the significant variation observed across the three columns of the figure.

Our discussion on how the MPM simulation handles different materials generalizes to the 3D scenario as well. The fluid spread remains consistent among different materials, yet the final saturation value differs. Per our prior explanation, this is expected. Materials with a lower porosity will naturally showcase higher values in the saturation. Per Table 5.1, claystone has the lowest porosity and softwood has the highest porosity among the materials of the four cubes. Thus, the wooden cube will have the lowest porosity value and, claystone, the highest.

The final output of the end-to-end pipeline, as visualised in Figure 5.18, reflects both the strengths and limitations of the proposed method. On one hand, a visualisation of fluid spread and wetness patterns across a three-dimensional scene has been successfully obtained. On the other hand, the resulting appearance may lack realism, which can be attributed to several factors. The quality of the representation produced by the preprocessing pipeline is itself constrained by the fidelity of the initial 3D Gaussian Splatting reconstruction; a low-quality Gaussian splat will inevitably yield a suboptimal simulation input. Furthermore, the realism of the fluid spread appearance is also limited by the absence of separate handling of albedo and specularities in the 3D Gaussian representation. Addressing this would require the incorporation of bidirectional reflectance distribution functions (BRDFs), which was out of the scope of the present work due to time constraints.

In terms of runtime, Table 5.2 highlights a time taken per simulation step that is directly proportional to the size of the MPM grid, as well as the number of input particles. This is expected; an increase in the number of grid nodes, as well as particles, will lead to additional computational overhead.

6.2. Limitations

The proposed approach is subject to several limitations. Firstly, due to the sequential, pipeline-based nature of the algorithm, failures in any individual component propagate to subsequent stages; should the preprocessing pipeline fail to remove Gaussian artifacts, for instance, such artifacts may negatively affect the results of the MPM simulation. This limitation becomes apparent upon visualisation of the final pipeline output, where a uniform horizontal grid of rain particles does not produce a correspondingly uniform distribution of fluid spread throughout the scene.

Secondly, achieving optimal results necessitates the careful tuning of multiple components; the preprocessing pipeline, in particular, must be calibrated to produce a satisfactory input for the MPM simulation. As a consequence, the parameter configurations yielding the best performance are liable to vary across different scenes.

Lastly, the MPM simulation itself is inherently limited in its capacity to accurately model the flow of fluid through porous media. Due to the constraints imposed by the 3DGS-based pipeline, as well as project time constraints, the implementation of a complex fluid-solid interaction was not feasible. Consequently, the implementation does not account for gravitational forces or the governing equations of incompressible fluid flow through porous media. The accumulation of water onto solid surfaces and puddle formation are also not handled in our framework. Furthermore, the encoding of fluid mass

onto the grid and the subsequent interpolation of these values by the particles introduces an inherent dependency on the MPM grid resolution, which may compromise the physical accuracy of the simulation results.

6.3. Reflection on research in a broader societal context

This study tackles the research gap of modelling adverse weather, more specifically the infiltration of rainwater into solid materials. As highlighted in Section 2, a range of works tackled physics simulations, yet only a couple of studies adapted Gaussian Splatting-based pipelines to adverse weather modeling. To the best of our knowledge, there has been no investigation on physics-based simulation of the interaction of rain with environments represented based on Gaussian primitives.

The prototype developed in this study is a proof of concept exploring the possibility of developing physically accurate simulations of adverse weather involving real-world photographic data. Extensions of this product can be employed to create large-scale adverse weather simulations with applications in agriculture and urban planning. For instance, the proposed model could potentially simulate the infiltration of rainwater into road embankments, a topic of great interest in geotechnical engineering [4]. Thus, this tool has implications and potential applications in multiple fields. It is crucial that this technology be developed and leveraged in a way that maximizes its utility and minimizes its risks, and as such it is important to understand its strengths and weaknesses.

The main strength of the developed tool is that of facilitating the visualization of fluid transfer from fluid to solid particles, as well as within solid objects. As such, the technology can be used to visualize rain passing through soil given a 3D Gaussian Splat that accurately represents the scene. On the other hand, one weakness of the method is that of handling flat surfaces. Oftentimes, these types of surfaces are represented by elongated and flatter Gaussians. As such, these Gaussians may be at larger distances from one another, or may be erroneously filtered out in the preprocessing pipeline. These errors could lead to an inaccurate representation of rain infiltration. For roads, we expect this issue to be easily solvable due to the ground infilling process (see Section 4.1). However, the application might fail in its current state to accurately study the infiltration of water into sloped terrain.

In its current state, the application may be unusable for the accurate modeling of large-scale infiltrations of water (such as landslides). However, we argue that this study provides the necessary groundwork to develop a tool that can fully tackle this challenge in the future.

Conclusion and Future Work

The purpose of this study was to investigate the feasibility of obtaining photorealistic rain simulations. Whereas existing works in the literature focused on appearance-based renditions of rain, our method was more physics-informed, approaching the topic of rainwater infiltration into solid media. To this end, 3DGS pipelines were considered the best representations to build on, due to their inherent compatibility with particle-based simulations. The output of a Gaussian Splat can thus be preprocessed into an input for a physics engine. Some good contenders to this end are engines based on the Material point Method (MPM), due to their satisfactory balance between performance and runtime. The MPM is also capable of simulating large-scale scenarios, thus being suitable for dense scenes.

One method in the literature, namely FeatureSplatting [9], already integrates an MPM engine into a 3DGS-based pipeline to run physics simulations. However, the focus of this method is per-object editing and simulation. Our method builds upon FeatureSplatting and extends it to a per-scene simulation of rainwater infiltration. We introduce a preprocessing pipeline that ignores Gaussian artifacts and adds per-object support particles. In the post-processing step, the Gaussian appearance parameters are tweaked to highlight per-particle saturation levels. The most notable contribution, however, is the extension of the MPM solver to handle the flow of fluid through solid objects without generating additional particles. By distributing fluid on an Eulerian grid, we enable particles to interpolate fluid mass from their neighbors, which creates smooth patterns of rainwater infiltration.

Due to its pipeline nature, the MPM simulation is constrained in some senses to ensure compatibility with the 3DGS representation. For instance, multi-phase, multi-point MPM engines were considered infeasible, despite their potential superior performance. In a multi-phase, multi-point MPM simulation, fluid would be treated as an independent series of particles interacting directly with the solid particles. One future direction could thus investigate the potential of running multi-point simulations, perhaps bypassing the appearance parameter update and treating the initial 3DGS representation as strictly an input to the MPM simulation.

As this study aimed to develop a proof of concept, the methodology incurs a few caveats related to the MPM simulation. The fluid flow schema has been simplified and adapted to our purposes, namely obtaining a simple measure of particle saturation that would inform the final Gaussian appearance. Consequently, aspects such as gravity and pore pressure were not taken into account. A notable extension to this study would thus implement a more physics-grounded fluid-solid coupling schema, taking into account, for instance, the flow of incompressible fluids through porous media as governed by Darcy's flux formula [15].

In addition, due to time constraints, in the appearance parameter update pipeline, we had to make an educated approximation of the Gaussian albedo as the 0-level Gaussian spherical harmonics. In practice, a more detailed approach would separate the Gaussian albedo and specularities and tune them separately for more photorealistic wetting patterns. An extension in that sense could associate Bidirectional Reflectance Distribution Functions (BRDFs) to each 3D Gaussian added to the simulation, which would allow for implementing a more advanced shader, producing better results.

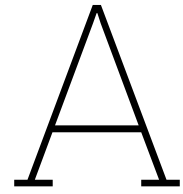
References

- [1] R. Morbidelli, C. Corradini, C. Saltalippi, A. Flammini, J. Dari, and R. S. Govindaraju, "Rainfall Infiltration Modeling: A Review," en, *Water*, vol. 10, no. 12, p. 1873, Dec. 2018, ISSN: 2073-4441. DOI: 10.3390/w10121873. [Online]. Available: <https://www.mdpi.com/2073-4441/10/12/1873>.
- [2] G. Markovič, M. Zeleňáková, D. Káposztásová, and G. Hudáková, "Rainwater infiltration in the urban areas," Ancona, Italy, May 2014, pp. 313–320. DOI: 10.2495/EID140271. [Online]. Available: <http://library.witpress.com/viewpaper.asp?pcode=EID14-027-1>.
- [3] R. W. Day, "Design and Repair for Surficial Slope Failures," en, *Practice Periodical on Structural Design and Construction*, vol. 1, no. 3, pp. 83–87, Aug. 1996, ISSN: 1084-0680, 1943-5576. DOI: 10.1061/(ASCE)1084-0680(1996)1:3(83). [Online]. Available: <https://ascelibrary.org/doi/10.1061/%28ASCE%291084-0680%281996%291%3A3%2883%29>.
- [4] J. Norambuena-Contreras, G. Arbat, P. García Nieto, and D. Castro-Fresno, "Nonlinear numerical simulation of rainwater infiltration through road embankments by FEM," en, *Applied Mathematics and Computation*, vol. 219, no. 4, pp. 1843–1852, Nov. 2012, ISSN: 00963003. DOI: 10.1016/j.amc.2012.08.025. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0096300312008181>.
- [5] 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," en, *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, Jan. 2022, ISSN: 0001-0782, 1557-7317. DOI: 10.1145/3503250. [Online]. Available: <https://dl.acm.org/doi/10.1145/3503250>.
- [6] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, *3D Gaussian Splatting for Real-Time Radiance Field Rendering*, arXiv:2308.04079 [cs], Aug. 2023. DOI: 10.48550/arXiv.2308.04079. [Online]. Available: <http://arxiv.org/abs/2308.04079>.
- [7] P. Borycki, W. Smolak, J. Waczyńska, M. Mazur, S. Tadeja, and P. Spurek, *GASP: Gaussian Splatting for Physic-Based Simulations*, arXiv:2409.05819 [cs], Oct. 2025. DOI: 10.48550/arXiv.2409.05819. [Online]. Available: <http://arxiv.org/abs/2409.05819>.
- [8] T. Xie et al., *PhysGaussian: Physics-Integrated 3D Gaussians for Generative Dynamics*, arXiv:2311.12198 [cs], Apr. 2024. DOI: 10.48550/arXiv.2311.12198. [Online]. Available: <http://arxiv.org/abs/2311.12198>.
- [9] R.-Z. Qiu, G. Yang, W. Zeng, and X. Wang, *Feature Splatting: Language-Driven Physics-Based Scene Synthesis and Editing*, Version Number: 1, 2024. DOI: 10.48550/ARXIV.2404.01223. [Online]. Available: <https://arxiv.org/abs/2404.01223>.
- [10] Q. Dai, X. Ni, Q. Shen, W. Chen, B. Chen, and M. Chu, *RainyGS: Efficient Rain Synthesis with Physically-Based Gaussian Splatting*, arXiv:2503.21442 [cs], Apr. 2025. DOI: 10.48550/arXiv.2503.21442. [Online]. Available: <http://arxiv.org/abs/2503.21442>.
- [11] G. Fiebelman, H. Averbuch-Elor, and S. Benaim, *Let it Snow! Animating Static Gaussian Scenes With Dynamic Weather Effects*, arXiv:2504.05296 [cs], Apr. 2025. DOI: 10.48550/arXiv.2504.05296. [Online]. Available: <http://arxiv.org/abs/2504.05296>.
- [12] R. Bridson, *Fluid Simulation for Computer Graphics*, en, 0th ed. A K Peters/CRC Press, Sep. 2015, ISBN: 978-1-315-26600-8. DOI: 10.1201/9781315266008. [Online]. Available: <https://tudelft.on.worldcat.org/oclc/963719787>.
- [13] C. Jiang, C. Schroeder, J. Teran, A. Stomakhin, and A. Selle, "The material point method for simulating continuum materials," en, in *ACM SIGGRAPH 2016 Courses*, Anaheim California: ACM, Jul. 2016, pp. 1–52, ISBN: 978-1-4503-4289-6. DOI: 10.1145/2897826.2927348. [Online]. Available: <https://dl.acm.org/doi/10.1145/2897826.2927348>.

- [14] D. Wang, B. Wang, W. Yuan, and L. Liu, "Investigation of rainfall intensity on the slope failure process using GPU-accelerated coupled MPM," en, *Computers and Geotechnics*, vol. 163, p. 105718, Nov. 2023, ISSN: 0266352X. DOI: 10.1016/j.compgeo.2023.105718. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0266352X23004755>.
- [15] T. Lenaerts, B. Adams, and P. Dutré, "Porous flow in particle-based fluid simulations," *ACM Transactions on Graphics (TOG)*, vol. 27, no. 3, pp. 1–8, 2008.
- [16] Y. Furukawa and C. Hernández, "Multi-View Stereo: A Tutorial," en, *Foundations and Trends® in Computer Graphics and Vision*, vol. 9, no. 1-2, pp. 1–148, 2015, ISSN: 1572-2740, 1572-2759. DOI: 10.1561/06000000052. [Online]. Available: <http://www.nowpublishers.com/article/Details/CGV-052>.
- [17] T. Samavati and M. Soryani, "Deep learning-based 3D reconstruction: A survey," en, *Artificial Intelligence Review*, vol. 56, no. 9, pp. 9175–9219, Sep. 2023, ISSN: 0269-2821, 1573-7462. DOI: 10.1007/s10462-023-10399-2. [Online]. Available: <https://link.springer.com/10.1007/s10462-023-10399-2>.
- [18] M. Chu et al., "Physics informed neural fields for smoke reconstruction with sparse data," en, *ACM Transactions on Graphics*, vol. 41, no. 4, pp. 1–14, Jul. 2022, ISSN: 0730-0301, 1557-7368. DOI: 10.1145/3528223.3530169. [Online]. Available: <https://dl.acm.org/doi/10.1145/3528223.3530169>.
- [19] Y. Feng, Y. Shang, X. Li, T. Shao, C. Jiang, and Y. Yang, *PIE-NeRF: Physics-based Interactive Elastodynamics with NeRF*, arXiv:2311.13099 [cs], Mar. 2024. DOI: 10.48550/arXiv.2311.13099. [Online]. Available: <http://arxiv.org/abs/2311.13099>.
- [20] X. Li et al., *PAC-NeRF: Physics Augmented Continuum Neural Radiance Fields for Geometry-Agnostic System Identification*, arXiv:2303.05512 [cs], Mar. 2023. DOI: 10.48550/arXiv.2303.05512. [Online]. Available: <http://arxiv.org/abs/2303.05512>.
- [21] W. Tan, A. Velasquez, and H. Song, *DEM-NeRF: A Neuro-Symbolic Method for Scientific Discovery through Physics-Informed Simulation*, arXiv:2507.21350 [cs], Jul. 2025. DOI: 10.48550/arXiv.2507.21350. [Online]. Available: <http://arxiv.org/abs/2507.21350>.
- [22] P. Makridis, "Gaussians as supervision for joint physical parameter estimation and appearance reconstruction of elastic objects," 2024.
- [23] T. Zhang et al., "PhysDreamer: Physics-Based Interaction with 3D Objects via Video Generation," en, in *Computer Vision – ECCV 2024*, A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, and G. Varol, Eds., vol. 15060, Series Title: Lecture Notes in Computer Science, Cham: Springer Nature Switzerland, 2025, pp. 388–406, ISBN: 978-3-031-72626-2 978-3-031-72627-9. DOI: 10.1007/978-3-031-72627-9_22. [Online]. Available: https://link.springer.com/10.1007/978-3-031-72627-9_22.
- [24] J. Xing, B. Wang, M. Chu, and B. Chen, "Gaussian Fluids: A Grid-Free Fluid Solver based on Gaussian Spatial Representation," en, in *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, Vancouver BC Canada: ACM, Aug. 2025, pp. 1–11, ISBN: 979-8-4007-1540-2. DOI: 10.1145/3721238.3730620. [Online]. Available: <https://dl.acm.org/doi/10.1145/3721238.3730620>.
- [25] "GaussFluids: Reconstructing Lagrangian Fluid Particles from Videos via Gaussian Splatting," 2025, Artwork Size: 13 pages Edition: 1286 ISBN: 9783038682950 Publisher: The Eurographics Association. DOI: 10.2312/PG.20251286. [Online]. Available: <https://diglib.eg.org/handle/10.2312/pg20251286>.
- [26] Y. Feng et al., *Gaussian Splashing: Unified Particles for Versatile Motion Synthesis and Rendering*, arXiv:2401.15318 [cs], Jul. 2024. DOI: 10.48550/arXiv.2401.15318. [Online]. Available: <http://arxiv.org/abs/2401.15318>.
- [27] Y. Xie et al., "FluidGS: Physics Informed Gaussian Splatting for Dynamic Fluid Reconstruction from Sparse Views," en, in *Proceedings of the 33rd ACM International Conference on Multimedia*, Dublin Ireland: ACM, Oct. 2025, pp. 8438–8447, ISBN: 979-8-4007-2035-2. DOI: 10.1145/3746027.3755500. [Online]. Available: <https://dl.acm.org/doi/10.1145/3746027.3755500>.

- [28] M. Rensen, M. Weinmann, B. Buschmann, and E. Eisemann, "Physics-informed gaussian splatting," Ph.D. dissertation, Master's thesis, 2024.
- [29] R. A. Gingold and J. J. Monaghan, "Smoothed particle hydrodynamics: Theory and application to non-spherical stars," en, *Monthly Notices of the Royal Astronomical Society*, vol. 181, no. 3, pp. 375–389, Dec. 1977, ISSN: 0035-8711, 1365-2966. DOI: 10.1093/mnras/181.3.375. [Online]. Available: <https://academic.oup.com/mnras/article-lookup/doi/10.1093/mnras/181.3.375>.
- [30] J. J. Monaghan, "Smoothed particle hydrodynamics," *Reports on Progress in Physics*, vol. 68, no. 8, pp. 1703–1759, Aug. 2005, ISSN: 0034-4885, 1361-6633. DOI: 10.1088/0034-4885/68/8/R01. [Online]. Available: <https://iopscience.iop.org/article/10.1088/0034-4885/68/8/R01>.
- [31] F. H. Harlow, "The marker-and-cell method," In *Von Karman Inst. for Fluid Dyn. Numerical Methods in Fluid Dyn. 38 p (SEE N79-27448 18-34)*, Jan. 1972, ADS Bibcode: 1972nmfd.vkifQ....H. [Online]. Available: <https://ui.adsabs.harvard.edu/abs/1972nmfd.vkifQ....H>.
- [32] X. Yan, C.-F. Li, X.-S. Chen, and S.-M. Hu, "MPM simulation of interacting fluids and solids," en, *Computer Graphics Forum*, vol. 37, no. 8, pp. 183–193, Dec. 2018, ISSN: 0167-7055, 1467-8659. DOI: 10.1111/cgf.13523. [Online]. Available: <https://onlinelibrary.wiley.com/doi/10.1111/cgf.13523>.
- [33] N. Wandel, M. Weinmann, M. Neidlin, and R. Klein, "Spline-PINN: Approaching PDEs without Data Using Fast, Physics-Informed Hermite-Spline CNNs," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, pp. 8529–8538, Jun. 2022, ISSN: 2374-3468, 2159-5399. DOI: 10.1609/aaai.v36i8.20830. [Online]. Available: <https://ojs.aaai.org/index.php/AAAI/article/view/20830>.
- [34] N. Wandel, M. Weinmann, and R. Klein, *Learning Incompressible Fluid Dynamics from Scratch – Towards Fast, Differentiable Fluid Models that Generalize*, Version Number: 3, 2020. DOI: 10.48550/ARXIV.2006.08762. [Online]. Available: <https://arxiv.org/abs/2006.08762>.
- [35] N. Wandel, M. Weinmann, and R. Klein, "Teaching the incompressible Navier–Stokes equations to fast neural surrogate models in three dimensions," en, *Physics of Fluids*, vol. 33, no. 4, p. 047117, Apr. 2021, ISSN: 1070-6631, 1089-7666. DOI: 10.1063/5.0047428. [Online]. Available: <https://pubs.aip.org/pof/article/33/4/047117/1065748/Teaching-the-incompressible-Navier-Stokes>.
- [36] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [37] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise view selection for unstructured multi-view stereo," in *European Conference on Computer Vision (ECCV)*, 2016.
- [38] A. Kirillov et al., "Segment anything," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 4015–4026.
- [39] M. Oquab et al., "Dinov2: Learning robust visual features without supervision," *arXiv preprint arXiv:2304.07193*, 2023.
- [40] Y. Hu et al., "DiffTaichi: Differentiable programming for physical simulation," *arXiv preprint arXiv:1910.00935*, 2019.
- [41] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., "A density-based algorithm for discovering clusters in large spatial databases with noise," in *kdd*, vol. 96, 1996, pp. 226–231.
- [42] R. A. Freeze and J. A. Cherry, *Groundwater*. Englewood Cliffs, New Jersey: Prentice-Hall, 1979, ISBN: 9780133653120.
- [43] K. V. Beard, V. Bringi, and M. Thurai, "A new understanding of raindrop shape," *Atmospheric Research*, vol. 97, no. 4, pp. 396–415, 2010, From the Lab to Models and Global Observations: Hans R. Pruppacher and Cloud Physics, ISSN: 0169-8095. DOI: <https://doi.org/10.1016/j.atmosres.2010.02.001>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0169809510000268>.

- [44] J. Lekner and M. C. Dorf, "Why some things are darker when wet," en, *Applied Optics*, vol. 27, no. 7, p. 1278, Apr. 1988, ISSN: 0003-6935, 1539-4522. DOI: 10.1364/AO.27.001278. [Online]. Available: <https://opg.optica.org/abstract.cfm?URI=ao-27-7-1278>.
- [45] S. A. Twomey, C. F. Bohren, and J. L. Mergenthaler, "Reflectance and albedo differences between wet and dry surfaces," en, *Applied Optics*, vol. 25, no. 3, p. 431, Feb. 1986, ISSN: 0003-6935, 1539-4522. DOI: 10.1364/AO.25.000431. [Online]. Available: <https://opg.optica.org/abstract.cfm?URI=ao-25-3-431>.
- [46] C. Xu, J. Kerr, and A. Kanazawa, *Splatfacto-W: A Nerfstudio Implementation of Gaussian Splatting for Unconstrained Photo Collections*, Version Number: 2, 2024. DOI: 10.48550/ARXIV.2407.12306. [Online]. Available: <https://arxiv.org/abs/2407.12306>.
- [47] Blender Online Community, *Blender 5.1 reference manual*, Blender Foundation, Amsterdam, Netherlands, 2026. [Online]. Available: <https://docs.blender.org/manual/en/latest/>.
- [48] M. Tancik et al., "Nerfstudio: A modular framework for neural radiance field development," in *ACM SIGGRAPH 2023 Conference Proceedings*, ser. SIGGRAPH '23, 2023.
- [49] L. F. Pires, A. C. Auler, W. L. Roque, and S. J. Mooney, "X-ray microtomography analysis of soil pore structure dynamics under wetting and drying cycles," en, *Geoderma*, vol. 362, p. 114 103, Mar. 2020, ISSN: 00167061. DOI: 10.1016/j.geoderma.2019.114103. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0016706119307037>.
- [50] A. Kienle, C. D'Andrea, F. Foschum, P. Taroni, and A. Pifferi, "Light propagation in dry and wet softwood," en, *Optics Express*, vol. 16, no. 13, p. 9895, Jun. 2008, ISSN: 1094-4087. DOI: 10.1364/OE.16.009895. [Online]. Available: <https://opg.optica.org/oe/abstract.cfm?uri=oe-16-13-9895>.
- [51] F. Zhang, S. Xie, D. Hu, J. Shao, and B. Gatmiri, "Effect of water content and structural anisotropy on mechanical property of claystone," en, *Applied Clay Science*, vol. 69, pp. 79–86, Nov. 2012, ISSN: 01691317. DOI: 10.1016/j.clay.2012.09.024. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0169131712002414>.
- [52] N. Ekin and O. Uyanik, "Anisotropy of reinforced concrete from geophysical methods," en, *Science of Sintering*, vol. 53, no. 3, pp. 323–345, 2021, ISSN: 0350-820X, 1820-7413. DOI: 10.2298/S0S2103323E. [Online]. Available: <https://doiserbia.nb.rs/Article.aspx?ID=0350-820X2103323E>.
- [53] M. Fukuda et al., "Determining Accurate Pore Structures of Polypropylene Membrane for ECMO Using FE-SEM Under Optimized Conditions," en, *Membranes*, vol. 15, no. 6, p. 174, Jun. 2025, ISSN: 2077-0375. DOI: 10.3390/membranes15060174. [Online]. Available: <https://www.mdpi.com/2077-0375/15/6/174>.
- [54] R. Tielen, E. Wobbles, M. Möller, and L. Beuth, "A High Order Material Point Method," en, *Procedia Engineering*, vol. 175, pp. 265–272, 2017, ISSN: 18777058. DOI: 10.1016/j.proeng.2017.01.022. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S187770581730022X>.



Reflection on use of Generative AI

In the making of this study, Generative AI was used for the following purposes:

- Resolving questions related to setting up Python libraries, such as NeRFStudio
- Resolving questions specific to the inner workings of Python libraries involved in the project, such as Taichi
- Automating the procedural capture of point clouds and Gaussian Splats from a specified camera angle
- Automating the creation of MPM simulation result plots
- Occasional rephrasing of report content for coherence and suitability