# POWERSIM: DIFFERENTIABLE PHYSICS SIMULATION AND RENDERING WITH POWER DIAGRAMS

**Anonymous authors**
Paper under double-blind review

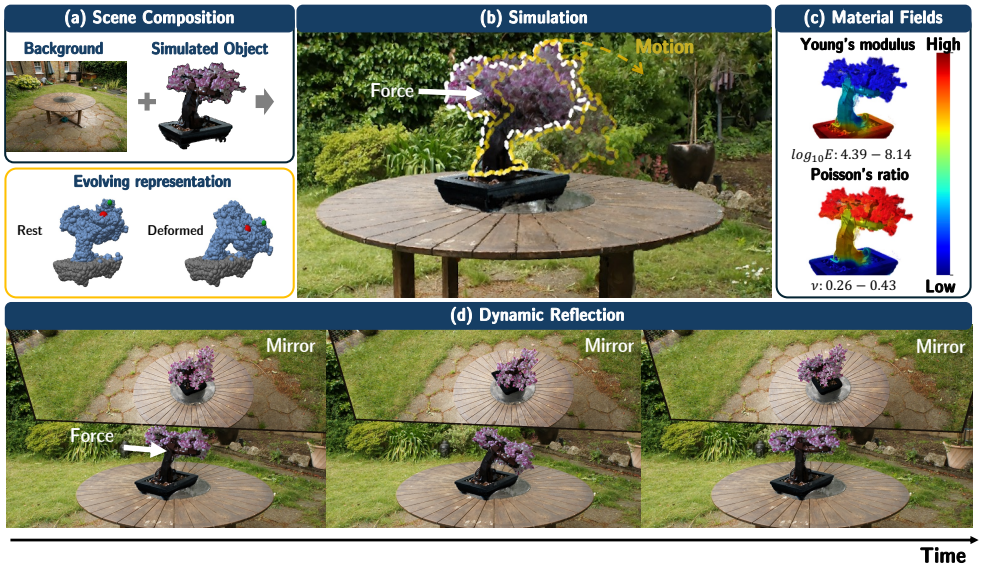


Figure 1: **PowerSim unifies MPM simulation and rendering with bounded power diagrams in an end-to-end differentiable pipeline.** (a) Objects from independent captures can be selected and composed into a scene. (b) Primitive geometry and appearance evolve under applied forces. (c) Spatially varying material properties can be estimated from video or assigned to control deformation. (d) Ray-traced reflections follow the object's motion and deformation. Our results are best viewed on our supplement website: `https://iclr14573.github.io/`

## ABSTRACT

We introduce **PowerSim**, a method to bring physically grounded, differentiable dynamics to PowerFoam's power diagram based 3D representation. PowerSim directly couples a pre-trained PowerFoam scene to the Material Point Method (MPM) by exploiting a natural alignment between the two: the geometric and appearance properties of each primitive correspond closely to the quantities MPM already tracks as an object deforms. Consequently, simulated motion can drive the scene's geometry and appearance directly, without an auxiliary representation in between. Built on this framework, we enable a range of applications on real and synthetic scenes: (1) simulating a static scene under user interaction, (2) recovering spatially varying material fields, (3) compositing primitives from independently captured scenes into a single simulation-ready scene and (4) ray-tracing reflections that update consistently as the object deforms. Our results suggest that PowerSim excels over previous frameworks for physically grounded dynamics, while unlocking unique advantages—such as secondary ray lighting effects on dynamic scenes.

## 1 INTRODUCTION

Radiance field methods have made 3D scene capture routine (Mildenhall et al., 2021; Kerbl et al., 2023). A handful of images are enough to render it from novel viewpoints. But without temporal data, they only give us one instant in time, and many interesting questions about a scene require us to infer what might happen next. For example, what would happen if we replaced the vase on the



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

table with a plant and pushed it? How would its reflection in a mirror change (Fig. 1)? What would happen if we pulled a loaf of bread apart (Fig. 2)? To answer these questions, we need a simulation that preserves the visual fidelity of the captured scene. To infer physical parameters from observed motion, we also need to make this simulation-and-rendering pipeline differentiable.

However, simulation and rendering place different demands on a scene representation. Continuum simulation requires material volumes and a treatment of contact and separation, while rendering requires geometry and appearance sufficient to generate images. Mesh-based pipelines start from explicit geometry for simulation and attach an appearance model to it (Mai et al., 2026; Held et al., 2026). Gaussian-based pipelines start from a representation designed for rendering and adapt it for simulation using the Material Point Method (MPM) (Xie et al., 2023).

But Gaussian mixtures do not explicitly specify non-intersecting volumes of materials, an explicit boundary surface or adjacency of primitives. PhysGaussian provides optional regularization of Gaussian anisotropy and filling of interior regions of objects with additional particles, which, however, do not lead to establishing explicit boundaries between primitives. Thus, specifying material volumes and masses, contacts, and refractions implies making some extra assumptions or using additional constructions beyond the Gaussian representation itself.

PowerFoam (Govindarajan et al., 2026) reconstructs a scene as bounded power-diagram cells with explicit boundaries and neighbors, providing geometry for both simulation and rendering. But deforming a partition is harder than moving a set of points. Moving one site changes neighboring cell boundaries, and large deformations can change cell adjacency. The challenge is to update the primitives and their neighborhoods together as the object deforms.

To address this, we introduce **PowerSim**, which treats each PowerFoam primitive as an MPM material point. MPM updates its position, while the deformation gradient supplies a rotation for its dipole plane and appearance directions, and an isotropic radius scale that matches the local volume change. We rebuild adjacency as the primitives move and change size, keeping cell boundaries consistent with the updated representation. The result is a differentiable pipeline from forces to pixels: we can simulate a captured scene under new forces and backpropagate through simulation and rendering to estimate spatially varying material properties from monocular deformation videos. PowerSim improves average dynamic reconstruction quality and Young's modulus estimation over the compared methods, and preserves more surface detail under large stretching and twisting deformations (Fig. 2). We also demonstrate elastic motion, compression, granular collapse, and multi-object interactions with reflections that follow the simulated motion. In summary, our contributions are:

1. PowerSim, a differentiable framework coupling PowerFoam with MPM through updates to primitive geometry, appearance, and adjacency, supporting large deformation and separation.
2. Estimation of spatially varying material fields from monocular deformation videos through differentiable simulation and rendering.
3. Optimization-free selection of object primitives from multi-view masks, enabling scene composition and multi-object simulation with ray-traced reflections.

## 2 Related Work

**Physics-based scene simulation.** A growing line of work couples reconstructed scenes to physics solvers. PhysGaussian (Xie et al., 2023) treats 3D Gaussians as MPM particles, using the same primitives for simulation and rendering. PhysDreamer (Zhang et al., 2024), DreamPhysics (Huang et al., 2024), and Physics3D (Liu et al., 2024) estimate material properties using generated videos or video diffusion priors. Spring-Gaus (Zhong et al., 2024) instead fits a spring-mass model to observed videos. Other representations include PAC-NeRF's voxel–particle coupling (Li et al., 2023) and PhysConvex's deformable convex primitives with reduced-order simulation (Wang et al., 2026). Pixie (Le et al., 2025) predicts material fields from visual features, while Vid2Sim (Chen et al., 2025) combines feed-forward reconstruction with optimization. PowerSim couples bounded power-diagram cells to MPM, updating their geometry, appearance, and adjacency during deformation.

**Rendering and scene representations.** 3DGRT and 3DGUT support ray tracing and secondary-ray effects with Gaussian representations (Moenne-Loccoz et al., 2024; Wu et al., 2025). PowerFoam (Govindarajan et al., 2026) supports both rasterization and ray tracing using bounded power-diagram cells. We build on this representation to simulate captured objects, estimate their material properties, and render reflections that follow their deformation.

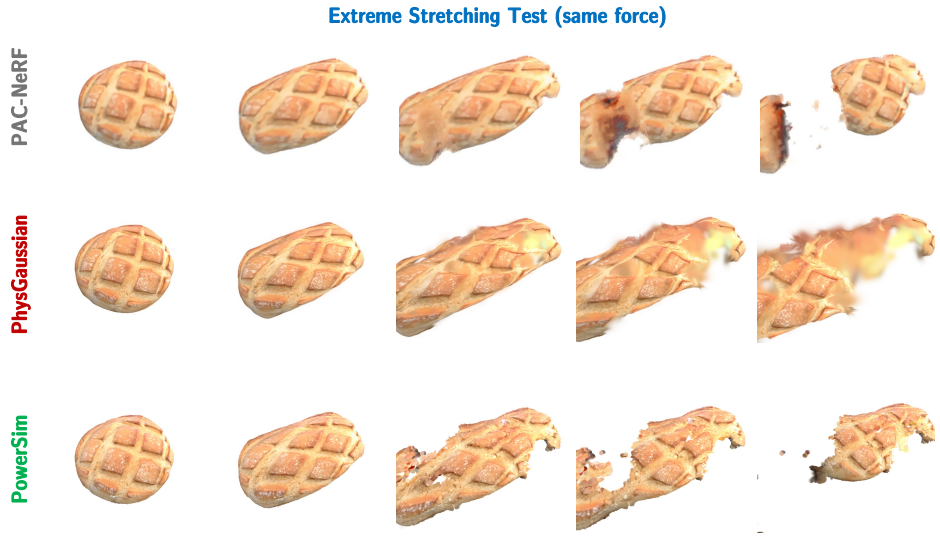


Figure 2: PowerSim partitions the reconstructed object volume into explicit cells, while PhysGaussian (Xie et al., 2023) uses Gaussian kernels that tend to concentrate near surfaces and optionally adds interior particles. Under extreme stretching with the same applied force, PAC-NeRF loses texture and develops abrupt cut-like boundaries, while PhysGaussian exhibits pronounced blurring in stretched regions. PowerSim retains more surface detail and sharper boundaries during separation.

## 3 PRELIMINARIES

**PowerFoam** (Govindarajan et al., 2026) represents a scene as bounded power-diagram cells. Each primitive has a center $\mathbf{p}_i \in \mathbb{R}^3$ and a scalar radius $\mathbf{r}_i$, which define its power cell:

$$\mathbf{P}_i = \left\{ \mathbf{x} \in \mathbb{R}^3 \;\middle|\; \|\mathbf{x} - \mathbf{p}_i\|^2 - \mathbf{r}_i^2 \le \|\mathbf{x} - \mathbf{p}_j\|^2 - \mathbf{r}_j^2,\ \forall j \right\}. \tag{1}$$

The cell is bounded by intersecting it with the ball $\mathbf{B}_i = \mathbf{B}(\mathbf{p}_i, \mathbf{r}_i)$. These cells provide explicit boundaries and neighbor relations.

*Geometry.* Each primitive contains a dipole plane through $\mathbf{p}_i$, with a material side of density $\boldsymbol{\sigma}_i$ and an empty side. A quaternion $\mathbf{q}_i$ defines the plane's orthonormal frame: normal $\mathbf{n}_i$ and tangents $\mathbf{t}_i, \mathbf{b}_i$. To represent surface detail, each plane carries $K$ detail sites with local in-plane coordinates $\mathbf{s}_{i,k} \in \mathbb{R}^2$ and normal displacements $d_{i,k} \in \mathbb{R}$. Their world positions are

$$\mathbf{x}_{i,k} = \mathbf{p}_i + \mathbf{r}_i\big(\mathbf{s}^1_{i,k}\mathbf{t}_i + \mathbf{s}^2_{i,k}\mathbf{b}_i\big) + d_{i,k}\mathbf{n}_i. \tag{2}$$

Interpolating the displacements gives a height field over the plane.

*Appearance.* Each detail site stores $L$ world-space appearance directions $\boldsymbol{\alpha}^{(l)}_{i,k}$ and corresponding RGB colors $\mathbf{c}^{(l)}_{i,k}$. The viewing direction determines how these colors are blended; a second interpolation blends colors across detail sites. We give both interpolation rules in Appendix A.2.

We write the reconstructed rest scene as $\mathcal{S}_0$, with image $I = \mathcal{R}(\mathcal{S}_0)$. For each primitive, $\mathbf{s}_i$, $d_i$, and $\boldsymbol{\alpha}_i$ collect its detail-site coordinates, displacements, and appearance directions. During simulation, we update $\{\mathbf{p}^t_i, \mathbf{r}^t_i, \mathbf{q}^t_i, \boldsymbol{\alpha}^t_i\}$ and keep the remaining attributes fixed. The local detail sites follow the primitive through Eq. 2, while the world-space appearance directions must be rotated explicitly.

**Material Point Method** (MPM) (Jiang et al., 2016) simulates a continuum body using particles and a background grid. Each particle carries position $\mathbf{x}_p$, velocity $\mathbf{v}_p$, mass $m_p$, deformation gradient $\mathbf{F}_p$, and an affine velocity matrix $\mathbf{C}_p$. Its material parameters are $\boldsymbol{\theta}_p = (E_p, \nu_p)$, where $E_p$ is Young's modulus and $\nu_p$ is Poisson's ratio. The deformation gradient tracks local rotation, stretch, and shear and determines stress through the material's constitutive model. Each step transfers particle mass and momentum to the grid, updates grid velocities under internal and external forces, and transfers the result back to the particles. Writing the simulator as $\mathcal{M}_{\boldsymbol{\theta}}$, one substep gives

$$\big(\mathbf{x}^{t+1}, \mathbf{v}^{t+1}, \mathbf{F}^{t+1}, \mathbf{C}^{t+1}\big) = \mathcal{M}_{\boldsymbol{\theta}}\big(\mathbf{x}^t, \mathbf{v}^t, \mathbf{F}^t, \mathbf{C}^t\big). \tag{3}$$

The full simulation loop is given in Appendix A.

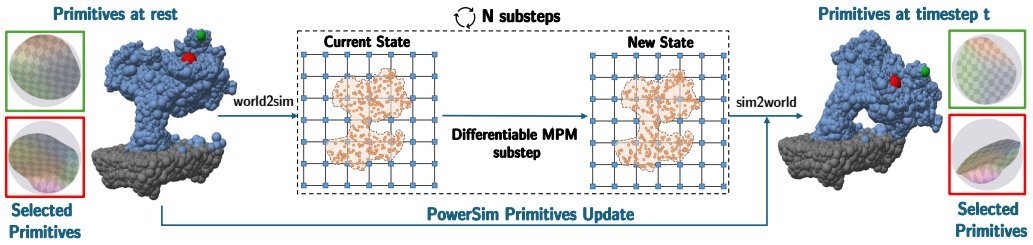


Figure 3: **PowerSim overview.** After $N$ MPM substeps, we update primitive positions, isotropically scale their radii to match local volume change, and rotate their dipole frames and appearance directions. We then rebuild adjacency for the updated primitives. Insets show two selected primitives before and after deformation.

# 4 METHOD

Given a pretrained PowerFoam scene, we update its primitives using MPM simulation output (§4.1; Fig. 3). We differentiate through simulation, primitive updates, and rendering to estimate spatially varying material properties from monocular deformation videos (§4.2). We also describe an optimization-free method to select simulation-ready object primitives from multi-view masks (§4.3).

## 4.1 POWERSIM: PHYSICS-GROUNDED POWERFOAM SIMULATION WITH MPM

A pretrained PowerFoam scene is purely static: every primitive carries geometry and appearance but no physical quantities, since nothing in photometric reconstruction needs them. In this work, we turn these primitives into continuum dynamic bodies by treating each primitive as an MPM particle and augmenting each with exactly the physical state MPM requires. Each augmented primitive thus carries geometry, appearance, and physical properties at once, and as it moves, all three must remain consistent with the physics-grounded simulation. The physical state $\mathbf{X}_t = \{\mathbf{x}_i^t, \mathbf{v}_i^t, \mathbf{F}_i^t, \mathbf{C}_i^t\}$ is advanced by MPM itself as discussed in Eq.3; what remains is to propagate these physical state update to the primitive channels. As shown in Fig.3, we derive an update operator $\mathcal{U}$ that, given MPM's simulation output at time $t$, maps a primitive's rest-state channels to their deformed versions:

$$\mathcal{U}: \ (\mathbf{p}_i^0,\ \mathbf{r}_i^0,\ \mathbf{q}_i^0,\ \boldsymbol{\alpha}_i^0) \ \longmapsto \ (\mathbf{p}_i^t,\ \mathbf{r}_i^t,\ \mathbf{q}_i^t,\ \boldsymbol{\alpha}_i^t), \tag{4}$$

**Volume and Mass.** We partition the MPM background grid into voxels of side length $\Delta x$ and divide each occupied voxel's volume $\Delta x^3$ equally among the particles it contains to obtain the rest volume $V_i^0$. Particle's mass is then $m_i = \rho V_i^0$ for a uniform density $\rho$ per scene. $V_i^0$ is computed once at rest, and subsequent volume change is carried by $J = \det(\mathbf{F})$.

**Primal Sites Update.** We let primitives' centers evolve by transforming their rest position $\mathbf{p}_i^0$ to MPM's own coordinate frame, $\mathbf{x}_i^0 = \Phi(\mathbf{p}_i^0)$; from there, MPM's loop runs through several $N$ substeps to evolve $\mathbf{x}_i^t$ directly, and we recover the primitive's world-space position at any time via the inverse map $\mathbf{p}_i^t = \Phi^{-1}(\mathbf{x}_i^t)$. Every other primitive property updates differently, by leveraging the local deformation information MPM tracks throughout the simulation: the **deformation gradient** $\mathbf{F}_i^t$, representing the primitive's accumulated deformation relative to its rest state. Physically, $\mathbf{F}_i^t$ encodes a mix of rotation, stretch, and shear accumulated in the material around primitive $i$. We first apply polar decomposition to $\mathbf{F}_i^t$ to obtain its rotation and stretch components:

$$\mathbf{F}_i^t = \mathbf{R}_i^t\,\mathbf{S}_i^t, \qquad \mathbf{F}_i^t = \mathbf{U}_i\boldsymbol{\Sigma}_i\mathbf{V}_i^\top, \ \ \mathbf{R}_i^t = \mathbf{U}_i\mathbf{V}_i^\top, \tag{5}$$

Here $\mathbf{R}_i^t$ is a rigid rotation and $\mathbf{S}_i^t$ a symmetric stretch, with singular values $\boldsymbol{\Sigma}_i = \mathrm{diag}(\sigma_i^{(1)}, \sigma_i^{(2)}, \sigma_i^{(3)})$. Next, we map $\mathbf{R}_i^t$ and $\mathbf{S}_i^t$ onto PowerFoam's remaining primitive channels.

**Power Radii Update.** PowerFoam's scalar radius has no anisotropic degree of freedom, so we project $\mathbf{S}_i^t$ onto the isotropic scale that reproduces the same local volume change:

$$\bar{\sigma}_i^t = \left(\sigma_i^{(1)}\sigma_i^{(2)}\sigma_i^{(3)}\right)^{1/3}, \qquad \mathbf{r}_i^t = \mathbf{r}_i^{(0)} \cdot \bar{\sigma}_i^t. \tag{6}$$

In this way, we deliberately leave each individual primitive to stay isotropic and compact, anisotropy arises from the developing geometry of the power diagram. On the other hand, the use of anisotropic

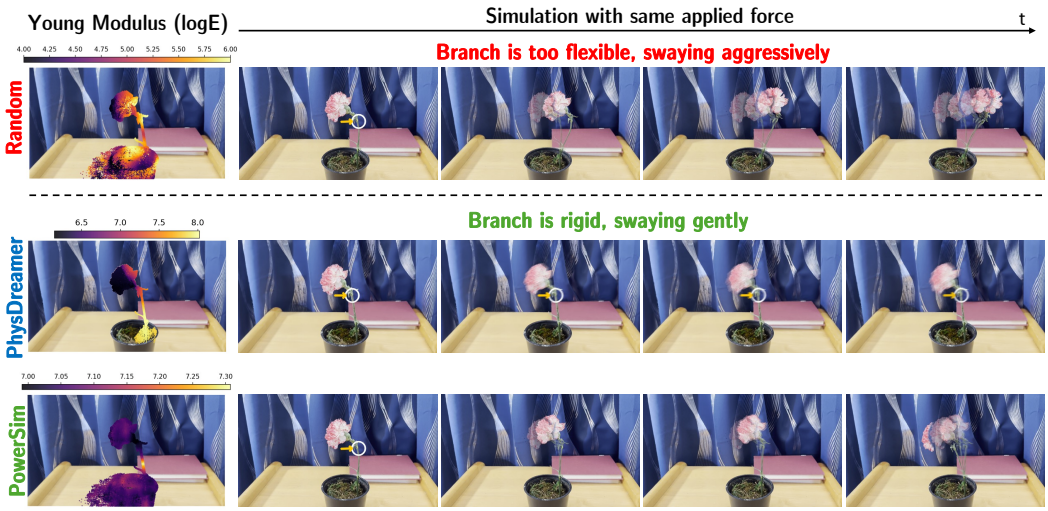


Figure 4: **Material estimation changes the simulated response.** Under the same applied force, random material assignment produces large stem bending, while PhysDreamer produces little bending. PowerSim produces back-and-forth motion with visible stem deformation. The left column shows Young's modulus; the remaining columns show matching simulation times.

3DGS may lead to absorbing deformation through stretching primitives, thus resulting in very elongated primitives and artifacts in the case of significant deformation as shown in Fig.2. PowerSim, however, offloads the problem of anisotropic shape changes to the geometry of neighboring cells, thus making the representation possible to be significantly deformed without elongating the primitives.

**Dipole Planes Update.** Dipole plane's normal and local reference frame $(\mathbf{n}_i, \mathbf{t}_i, \mathbf{b}_i)$ are parameterized by a single quaternion $\mathbf{q}_i^0$. We update $\mathbf{q}_i^0$ correspondingly with the rotation $\mathbf{R}_i^t$ obtained from the polar decomposition, since this naturally rotates the primitive's geometry to the corresponding angle without disturbing the relative orthogonality of $\mathbf{n}_i, \mathbf{t}_i, \mathbf{b}_i$. The quaternion update is achieved via

$$\mathbf{q}_i^t = \mathbf{q}_i^0 \otimes \Delta\mathbf{q}_i^t, \qquad \Delta\mathbf{q}_i^t = \mathrm{quat}\big(\mathbf{R}_i^{t\top}\big), \tag{7}$$

composed with the rest orientation $\mathbf{q}_i^0$ as the left operand. Note that the normal and tangents are encoded as row vectors of the rotation matrix, which is what leads to the order of multiplication and to composing with the *inverse* (transpose) of $\mathbf{R}_i^t$.

**Detail Sites Update.** From §3, each dipole plane carries $K$ detail sites with local coordinates $\mathbf{s}_i$, displacement $d_i$, and radiance axis $\boldsymbol{\alpha}_i$. The first two are expressed relative to the primal site's position, radius, and frame, so they need no update: re-evaluating Eq. 2 against the deformed primal site places them correctly for free. The radiance axis is a world-space direction and inherits nothing from the frame, so we update it explicitly, rotating the axis while holding its color coefficients fixed. We require that a co-rotating viewer sees no change: for every world direction $\mathbf{d}$, the shading response must match what the rest-pose primitive produces for the back-rotated direction $\mathbf{R}_i^{t\top}\mathbf{d}$,

$$\big\|\mathbf{d} - \boldsymbol{\alpha}_i^t\big\|_2 = \big\|\mathbf{R}_i^{t\top}\mathbf{d} - \boldsymbol{\alpha}_i^0\big\|_2, \tag{8}$$

which, by orthogonality of $\mathbf{R}_i^t$, gives

$$\boldsymbol{\alpha}_i^t = \mathbf{R}_i^t \boldsymbol{\alpha}_i^0. \tag{9}$$

**Topology Update.** Since primitives change every step, neighbor relations are created and broken as the object deforms, and a stale topology would clip cells against the wrong neighbors. We therefore rebuild the adjacency after every update as the Čech complex of the primitives' bounding spheres, which is the same structure PowerFoam already uses for rendering.

### 4.2 Material Field Optimization

Coupled with MPM, PowerFoam primitives evolve under physically grounded rules; what remains is to determine the material that drives them. Their effect is not subtle: Fig. 4 contrasts the unrealistic motion produced by a wrong assignment (top row) with the plausible deformation obtained from correct parameters (third row). Manually assigning them, however, demands expert knowledge,

does not scale beyond a handful of hand-tuned demo scenes, and is simply unavailable for objects reconstructed from video alone. We instead recover them directly via optimization from an observed video of the object deforming, after which new forces can be applied for new simulation.

**Problem setup.** Given a reference video $\{I_t\}_{t=1}^{T}$ of an object in motion, which is captured directly or synthesized by a video generation model, our goal is to estimate a spatial material field, $\boldsymbol{\theta}_i = \{\theta_i\}_{i=1}^{N}$, where $\boldsymbol{\theta}_i = (E_i, \nu_i)$ indicates the Young's modulus and Poisson's ratio, such that PowerSim's simulated results reproduce the observed motion. Because the impulse setting the object in motion is generally unobserved, we optimize each primitive's initial velocity $\mathbf{v}_i^0$ alongside the material.

**Optimization.** At each training iteration, we run the full forward pipeline – MPM simulation, our primitive-channel update, and rendering – to obtain a sequence of simulated frames. Here we abuse $t$ to index video frames rather than substeps: advancing from frame $t$ to $t{+}1$ takes $N_{\text{sub}} = \Delta t_{\text{frame}}/\Delta t_{\text{sub}}$ MPM substeps, so one simulated frame of the pipeline is

$$\hat{I}_{t+1} = \big(\mathcal{R} \circ \mathcal{U} \circ \mathcal{M}_{\boldsymbol{\theta}}^{N_{\text{sub}}}\big)(\mathbf{X}_t), \tag{10}$$

where $\mathcal{M}_{\boldsymbol{\theta}}^{N_{\text{sub}}}$ advances the physical state to $\mathbf{X}_{t+1}$, $\mathcal{U}$ maps it – with the rest state $\mathcal{S}_0$ held fixed – to the primitive state $\mathcal{S}_{t+1} = \mathcal{U}(\mathcal{S}_0, \mathbf{X}_{t+1})$, and $\mathcal{R}$ renders it. The rollout starts from rest, $\mathbf{X}_0 = \big(\Phi(\mathbf{p}^0),\ \mathbf{v}^0,\ \mathbf{I},\ \mathbf{0}\big)$. Because $\mathcal{R}, \mathcal{U}$, and $\mathcal{M}$ are each differentiable, so is Eq. 10. To update neighbors, we follow similarly to PowerFoam's training and rebuild adjacency at every frame from detached copies of the deformed sites. Then the rendering uses the original gradient-carrying sites with the updated adjacency list, neighborhood relations determine cell clipping without contributing gradients. This allows photometric losses on rendered frames to backpropagate to the unknowns. We compare each simulated frame against its reference video under:

$$\mathcal{L}(\hat{I}_t, I_t) = (1-\lambda)\,\|\hat{I}_t - I_t\|_2^2 + \lambda\,\big(1 - \text{SSIM}(\hat{I}_t, I_t)\big), \tag{11}$$

so that $\partial\mathcal{L}/\partial\boldsymbol{\theta}$ and $\partial\mathcal{L}/\partial\mathbf{v}^0$ pass from the rendered pixels through $\mathcal{R}$ and $\mathcal{U}$ into the MPM rollout, and the unknowns are updated by the gradient descent. Following PhysDreamer (Zhang et al., 2024), we apply truncated backpropagation-through-time (BPTT) to avoid gradient explosion/vanishing. Rather than fitting $\boldsymbol{\theta}$ and $\mathbf{v}^0$ jointly, we optimize in two stages, recovering the initial velocity first and the material afterwards (more details in §A).

### 4.3 Dynamic Primitives Selection via Render-weighted voting

A reconstructed real-world scene rarely contains a single object: a single kitchen scene contains thousands of primitives covering different objects, yet a simulation concerns one of them. This requires identifying the simulated primitives and assigning them material properties $\boldsymbol{\theta}_i$, leaving the rest as static geometry that is rendered but not simulated. However, curating these primitives by hand in 3D is tedious, and hence we tackle this issue with a simple method that leverages multi-view 2D segmentation masks at test-time, making it natural for picking primitives ready for simulation.

**Render-weighted voting.** The user specifies the object through 2D segmentation masks $\{M_j\}_{j\in\mathcal{J}}$ over a subset $\mathcal{J}$ of the training views, where $M_j(u) \in \{0,1\}$ marks whether pixel $u$ belongs to the object; off-the-shelf models can easily produce these from a text prompt or a click (Kirillov et al., 2023; Liu et al., 2023; Cheng et al., 2023). We seek a per-primitive label $\ell_i \in \{0,1\}$ whose selected primitives, rendered alone, reproduce $M_j$ on the annotated views and stay consistent on the rest. SemanticFoam (Sharafeldin et al., 2026) solves this problem by optimization and supervises the rendered feature against the masks; we instead read it off a trained checkpoint, since PowerFoam's rasterizer composites any per-primitive quantity the same way it composites color. Replacing colors with scalar features $f_i$, the rendered feature at pixel $u$ of view $j$ is

$$\hat{F}_j(u) = \sum_{i=1}^{N} w_{i,j}(u)\, f_i, \tag{12}$$

where $w_{i,j}(u)$ is the compositing weight of primitive $i$ along the ray through $u$; which is the weight with which its color reaches the pixel. A primitive's membership then follows from how much of its rendered contribution falls inside the masks versus outside, accumulated over views:

$$s_i^+ = \sum_{j\in\mathcal{J}}\sum_{u} w_{i,j}(u)\, M_j(u), \qquad s_i^- = \sum_{j\in\mathcal{J}}\sum_{u} w_{i,j}(u)\,\big(1 - M_j(u)\big), \tag{13}$$

We select primitive $i$, setting $\ell_i = 1$, when $s_i^+ > \beta\, s_i^-$, with $\beta < 1$ discounting the outside vote since masks tend to miss thin structures (We provide ablation of $\beta$ in §A.5).




Figure 5: **Scene composition.** Our optimization-free primitive selection enables removing the vase from a captured scene (a–b) and selecting a plant from another capture (c, green). We align and insert the selected primitives to create a scene ready for simulation (d).

Computing $\{s_i^+, s_i^-\}$ never requires the weights themselves. Since Eq. 12 is linear in $f$, the vote of primitive $i$ in view $j$ is a derivative of the rendered feature image:

$$\sum_u w_{i,j}(u)\, M_j(u) = \frac{\partial}{\partial f_i} \sum_u \hat{F}_j(u)\, M_j(u). \tag{14}$$

We therefore render the feature image, sum it over the masked pixels, and differentiate that scalar with respect to $f$: the gradient on $f_i$ is exactly primitive $i$'s vote, so one backward pass per view yields all $N$ votes at once, and $s_i^-$ follows with $1 - M_j$. Primitives no view observes ($w_{i,j}(u) = 0$ everywhere, hence $s_i^+ = s_i^- = 0$) inherit the majority label of their power-diagram neighbors, using the adjacency the renderer already maintains. The procedure needs no training and no checkpoint change, and handles several objects by voting one score per class and taking the arg-max.

**From selection to simulation and editing.** The selected primitives become material points, receiving $\boldsymbol{\theta}_i$ — a user-specified material or a field recovered by §4.2 — while the rest stay fixed and are rendered alongside them. The selection also supports editing. Removal deletes the selected primitives and rebuilds the power-diagram adjacency as shown in Fig.5. Insertion brings cropped primitives into a target scene of unrelated frame and scale. The two are aligned by a simple transform: rotation, translation, and scaling, which is followed by adjacency rebuilt over the merged set. Replacement is removal followed by insertion; Fig.5 shows the vase replaced.

## 5 Experimental Results

We evaluate simulation quality and material estimation against prior physics-coupled pipelines. We also demonstrate varied object dynamics (Fig. 10) and secondary-ray effects in dynamic scenes (Fig. 8). Primitive selection is evaluated in §A.5. We provide more simulation results in §A.

**Benchmarks and Metrics.** To assess both physics simulation and material estimation, we use Vid2Sim benchmark (Chen et al., 2025) which includes 12 objects from GSO (Google Scanned Object) (Downs et al., 2022), each dropped under gravity and simulated with FEM with known ground-truth material parameters. We evaluate dynamic reconstruction quality against the reference videos via PSNR and SSIM (Wang et al., 2004), and material estimation by the absolute error of the recovered $\log_{10} E$ and $\nu$.

**Baselines.** For physics simulation and material estimation, we compare against representative physics-coupled pipelines on other representations: PhysDreamer (Zhang et al., 2024) on 3DGS, and PAC-NeRF (Li et al., 2023) on NeRF. PhysDreamer extends PhysGaussian (Xie et al., 2023) with additional material optimization. For 3D segmentation, we compare against Semantic Foam (Sharafeldin et al., 2026), which requires optimization to obtain a per-primitive semantic field on the same foam family, and against the Gaussian-based Gaussian Grouping (Ye et al., 2024), and LabelGS (Zhang et al., 2025), with numbers taken from (Sharafeldin et al., 2026).

**Dynamic Reconstruction Evaluation.** PowerSim achieves the highest mean PSNR and SSIM across the 12 objects (Tab. 1). It leads in PSNR on 10 objects, reaching 25.73 dB on average compared with 22.06 dB for PAC-NeRF and 19.00 dB for PhysDreamer. SSIM gains are smaller, with a mean of 0.930 versus 0.924 and 0.908, respectively. Fig. 6 highlights PowerSim's preservation of texture and object shape during motion, across both the deformable backpack and the nearly rigid Mario figure. Baselines exhibit texture blurring, shape distortion, or appearance artifacts.

**Material Estimation Evaluation.** PowerSim achieves the lowest mean MAE in $\log(E)$, reducing the error from 0.60 for PhysDreamer to 0.44, a 27% reduction (Tab. 2). It achieves the lowest error on




Figure 6: **Dynamic reconstruction under deformation and contact.** PowerSim preserves the backpack's texture and Mario's shape as they fall and contact the ground. PAC-NeRF has blurred textures and distorted silhouettes, while PhysDreamer shows discrepancies in deformation and appearance. Reference frames are shown in the bottom row; time progresses from left to right.

Table 1: **Dynamic reconstruction on 12 objects.** PowerSim achieves the highest PSNR on 10 objects and SSIM on seven. Bold and underline indicate the best and second-best results, respectively.

| Metrics | Method | backpack | bell | blocks | bus | cream | elephant | grandpa | leather | lion | mario | sofa | turtle | Mean |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PSNR ↑ | PAC-NeRF | 19.37 | **25.00** | 23.36 | 20.72 | 23.24 | 22.27 | 21.63 | 20.85 | 22.66 | **21.01** | 22.49 | 22.19 | 22.06 |
| | PhysDreamer | 18.93 | 19.54 | 19.79 | 18.82 | 19.83 | 17.14 | 16.91 | 18.27 | 18.28 | 17.56 | 19.58 | 23.41 | 19.00 |
| | Ours | **23.69** | 24.39 | **29.56** | **26.84** | **25.82** | **22.80** | **22.01** | **35.33** | **25.17** | 20.31 | **24.72** | **28.11** | **25.73** |
| SSIM ↑ | PAC-NeRF | 0.887 | **0.956** | 0.940 | 0.908 | 0.893 | **0.922** | **0.939** | 0.932 | **0.936** | 0.921 | **0.926** | 0.923 | 0.924 |
| | PhysDreamer | 0.856 | 0.940 | 0.917 | 0.909 | 0.876 | 0.892 | 0.922 | 0.945 | 0.910 | 0.901 | 0.894 | 0.939 | 0.908 |
| | Ours | **0.889** | 0.944 | **0.953** | **0.938** | **0.923** | 0.908 | 0.889 | **0.978** | 0.930 | **0.941** | 0.918 | **0.953** | **0.930** |

seven of the 12 objects, with particularly large gains on blocks, lion, and turtle. For Poisson's ratio, PowerSim matches PhysDreamer's mean MAE of 0.16 at the reported precision.

Table 2: **Material estimation on 12 objects.** MAE in log Young's modulus and Poisson's ratio (lower is better). Bold and underline indicate the best and second-best results, respectively.

| Metrics | Method | backpack | bell | blocks | bus | cream | elephant | grandpa | leather | lion | mario | sofa | turtle | Mean |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\log(E)$ | PAC-NeRF | 3.28 | 1.08 | 4.02 | 3.30 | 3.22 | 3.05 | 2.99 | 1.20 | 2.34 | 3.37 | **0.20** | 1.94 | 2.50 |
| | PhysDreamer | 0.10 | 1.07 | 0.74 | **0.40** | **0.28** | 0.54 | **0.43** | **0.33** | 1.01 | 1.18 | 0.68 | 0.44 | 0.60 |
| | **Ours** | **0.06** | **0.81** | **0.16** | 0.60 | 0.37 | **0.36** | 0.47 | 1.21 | **0.20** | **0.60** | 0.33 | **0.09** | **0.44** |
| $\nu$ | PAC-NeRF | 0.21 | 0.23 | 0.33 | 0.16 | **0.12** | 0.06 | 0.36 | 0.26 | 0.14 | 0.33 | 0.30 | **0.01** | 0.21 |
| | PhysDreamer | 0.26 | **0.01** | **0.10** | 0.14 | 0.14 | **0.04** | 0.33 | **0.11** | **0.06** | 0.44 | **0.05** | 0.29 | **0.16** |
| | **Ours** | **0.17** | 0.18 | 0.16 | **0.12** | 0.17 | 0.27 | **0.01** | **0.11** | 0.16 | **0.25** | 0.09 | 0.21 | **0.16** |

**Ablation Studies.** We test role of geometry and appearance updates under large twisting and upward pulling (Fig. 7). Keeping dipole orientations fixed produces ragged boundaries and protruding surface fragments. Keeping appearance directions fixed introduces color inconsistencies as the surface rotates. Full update preserves cleaner boundaries and more consistent surface appearance throughout the motion. Under the same loading conditions, PhysGaussian shows blurred textures and fragmented stretched regions.

**Additional Applications.** With PowerSim, users can combine multiple objects into a single background scene and simulate all at once. Additionally, PowerSim supports both ray tracing and rasterization, opening up interesting application for physics simulation under complex secondary ray tracing effects. As demonstrated in Fig. 8, our framework successfully handles multi-object interactions, capturing dynamic physics simulations and their reflections in a mirror environment.

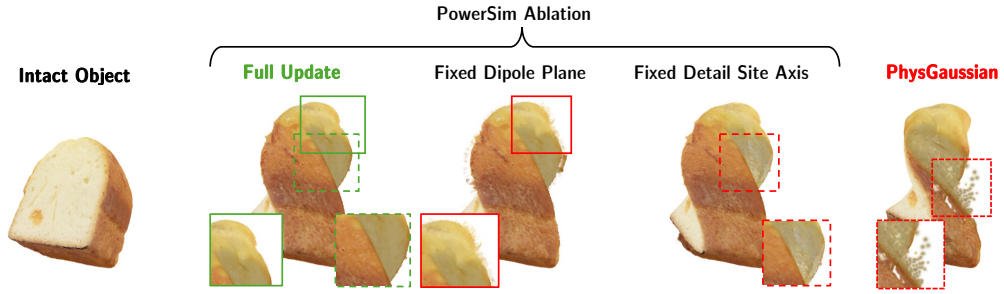


Figure 7: **Geometry and appearance updates under large deformation.** Fixing dipole orientations produces ragged boundaries, while fixing appearance directions introduces color inconsistencies. The full PowerSim update preserves cleaner surface detail; PhysGaussian shows blurring and fragmentation in stretched regions. Insets highlight these differences. Best viewed on supplementary webpage.

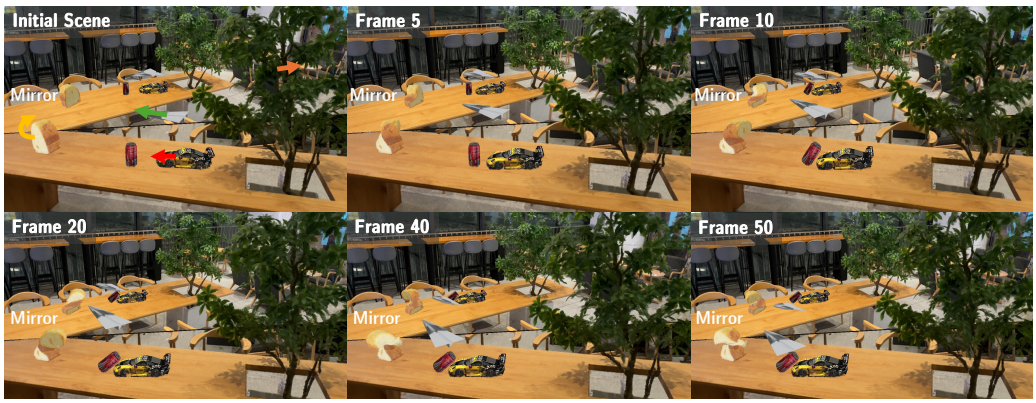


Figure 8: **Ray tracing on dynamic scenes.** Independently captured objects are composited and simulated together, with mirror reflections that follow their motion and deformation.




Figure 9: **Disocclusion artifacts.** PowerSim can select and move the telephone handset, but its motion exposes poorly reconstructed regions, revealing gaps and visual artifacts.

## 6 Discussion

PowerSim shows how a structured scene representation can support both physics simulation and rendering as objects deform. Updating primitive geometry, appearance and adjacency allows captured scenes to respond to new forces while retaining surface detail and supporting dynamic reflections. Differentiation through simulation and rendering also enables material estimation from deformation videos. Our coupling remains an approximation. Scalar radii capture isotropic scale changes, while dipole frames and appearance directions follow local rotation; individual primitives do not reproduce the full stretch and shear of the continuum. Although neighboring cells change shape as their sites move, this does not guarantee exact agreement between cell volumes and simulated material volumes. A richer deformation model could improve this correspondence. The results also depend on the captured geometry. Our selection method can isolate parts such as a telephone handset without optimization, but moving them may expose regions poorly observed during capture, revealing gaps and artifacts (Fig. 9). Generative priors could help complete these regions, which we leave to future work. Material estimation has a related ambiguity: different combinations of stiffness, initial velocity, and physical assumptions can explain similar image motion. The recovered parameters should therefore be interpreted under the assumed density, scale, and boundary conditions.

### AI USE STATEMENT

In this work, we used generative AI tools for polishing and translating the manuscript text, producing initial drafts of some passages, condensing related literature, and configuring the software environments. The core research, including the research questions, the coupling between PowerFoam and MPM, the experimental design, and the implementation of PowerSim, was carried out by the authors without AI assistance; theoretical and proof-related uses do not apply to this paper. Every AI-assisted contribution was checked by the authors: generated text was revised by hand, environment setup scripts were validated by executing them and examining their outputs, and literature summaries were compared against the cited papers. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

### ETHICS STATEMENT

This research uses only publicly available 3D assets, benchmark scenes, and captured object data, and involves no human participants or personally identifiable information.

### REPRODUCIBILITY STATEMENT

Section 4.1 defines the full primitive update operator (positions, radii, dipole frames, appearance directions, and adjacency), Section 4.2 describes the differentiable rollout and loss, and Section 4.3 gives the render-weighted voting procedure for primitive selection. Section A.3 details the MPM simulation loop, Section A.4 lists the material-field architecture, optimizer settings, and two-stage training schedule, and Tab. 3 specifies the constitutive model for every scene. Evaluation uses the public Vid2Sim benchmark and Mip-NeRF 360 scenes, and all external assets are listed in Section A.6. Code will be released upon publication.

## REFERENCES

Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5460–5469. IEEE, 2022.

Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16123–16133, 2022.

Chuhao Chen, Zhiyang Dou, Chen Wang, Yiming Huang, Anjun Chen, Qiao Feng, Jiatao Gu, and Lingjie Liu. Vid2sim: Generalizable, video-based reconstruction of appearance, geometry and physics for mesh-free simulation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 26545–26555, 2025.

Yangming Cheng, Liulei Li, Yuanyou Xu, Xiaodi Li, Zongxin Yang, Wenguan Wang, and Yi Yang. Segment and track anything. *arXiv preprint arXiv:2305.06558*, 2023.

Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. In *2022 international conference on robotics and automation (ICRA)*, pp. 2553–2560. Ieee, 2022.

Shrisudhan Govindarajan, Daniel Rebain, Dor Verbin, Kwang Moo Yi, Anish Prabhu, and Andrea Tagliasacchi. Power foam: Unifying real-time differentiable ray tracing and rasterization. *arXiv*, 2026.

Jan Held, Sanghyun Son, Renaud Vandeghen, Daniel Rebain, Matheus Gadelha, Yi Zhou, Anthony Cioppa, Ming C. Lin, Marc Van Droogenbroeck, and Andrea Tagliasacchi. MeshSplatting: Differentiable Rendering with Opaque Meshes. In *Computer Vision and Pattern Recognition (CVPR)*, 2026. URL https://arxiv.org/abs/2512.06818. oral.

Tianyu Huang, Yihan Zeng, Hui Li, Wangmeng Zuo, and Rynson WH Lau. Dreamphysics: Learning physical properties of dynamic 3d gaussians with video diffusion priors. *arXiv preprint arXiv:2406.01476*, 2024.

Chenfanfu Jiang, Craig Schroeder, Andrew Selle, Joseph Teran, and Alexey Stomakhin. The affine particle-in-cell method. *ACM Transactions on Graphics (TOG)*, 34(4):1–10, 2015.

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

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, George Drettakis, et al. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023.

Long Le, Ryan Lucas, Chen Wang, Chuhao Chen, Dinesh Jayaraman, Eric Eaton, and Lingjie Liu. Pixie: Fast and generalizable supervised learning of 3d physics from pixels. *arXiv preprint arXiv:2508.17437*, 2025.

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

Fangfu Liu, Hanyang Wang, Shunyu Yao, Shengjun Zhang, Jie Zhou, and Yueqi Duan. Physics3d: Learning physical properties of 3d gaussians via video diffusion. *arXiv preprint arXiv:2406.04338*, 2024.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023.

Alexander Mai, Trevor Hedstrom, George Kopanas, Janne Kontkanen, Falko Kuester, and Jonathan T Barron. Radiance meshes for volumetric reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8267–8277, 2026.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

Nicolas Moenne-Loccoz, Ashkan Mirzaei, Or Perel, Riccardo De Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler, Nicholas Sharp, and Zan Gojcic. 3d gaussian ray tracing: Fast tracing of particle scenes. *ACM Transactions on Graphics (TOG)*, 43(6):1–19, 2024.

Amr Sharafeldin, Shrisudhan Govindarajan, Thomas Walker, Aryan Mikaeili, Daniel Rebain, Kwang Moo Yi, and Andrea Tagliasacchi. Semantic foam: Unifying spatial and semantic scene decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Dan Wang, Xinrui Cui, Serge Belongie, and Ravi Ramamoorthi. Physconvex: Physics-informed 3d dynamic convex radiance fields for reconstruction and simulation, 2026. URL `https://arxiv.org/abs/2602.18886`.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Qi Wu, Janick Martinez Esturo, Ashkan Mirzaei, Nicolas Moenne-Loccoz, and Zan Gojcic. 3dgut: Enabling distorted cameras and secondary rays in gaussian splatting. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26036–26046. IEEE, 2025.

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

Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes. In *European conference on computer vision*, pp. 162–179. Springer, 2024.

Tianyuan Zhang, Hong-Xing Yu, Rundi Wu, Brandon Y. Feng, Changxi Zheng, Noah Snavely, Jiajun Wu, and William T. Freeman. PhysDreamer: Physics-based interaction with 3d objects via video generation. In *European Conference on Computer Vision*. Springer, 2024.

Yupeng Zhang, Dezhi Zheng, Ping Lu, Han Zhang, Lei Wang, Liping Xiang, Cheng Luo, Kaijun Deng, Xiaowen Fu, Linlin Shen, et al. Labelgs: Label-aware 3d gaussian splatting for 3d scene segmentation. In *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*, pp. 47–61. Springer, 2025.

Licheng Zhong, Hong-Xing Yu, Jiajun Wu, and Yunzhu Li. Reconstruction and simulation of elastic objects with spring-mass 3d gaussians. In *European Conference on Computer Vision (ECCV)*, 2024.

# A Appendix

## A.1 Project Website

We provide link to our project website at `https://iclr14573.github.io/`.

## A.2 PowerFoam Geometry and Appearance Details

We provide the interpolation rules used by the PowerFoam representation introduced in Section 3. These rules are inherited from PowerFoam (Govindarajan et al., 2026).

**Surface interpolation.** The dipole plane separates the cell into a material side opposite its normal $\mathbf{n}_i$, with density $\boldsymbol{\sigma}_i \in \mathbb{R}_+$, and an empty side with zero density. The detail-site displacements bend this plane into a height field. For an in-plane query $\bar{\mathbf{x}} \in \mathbb{R}^2$, expressed in the same radius-normalized coordinates as $\mathbf{s}_{i,k}$, the weight of detail site $k$ is

$$w_{i,k}(\bar{\mathbf{x}}) = \exp(-\tau \|\bar{\mathbf{x}} - \mathbf{s}_{i,k}\|_2)\,, \tag{15}$$

where $\tau$ controls the sharpness of the interpolation. The interpolated normal displacement is

$$h_i(\bar{\mathbf{x}}) = \frac{\sum_{k=1}^{K} w_{i,k}(\bar{\mathbf{x}})\, d_{i,k}}{\sum_{k=1}^{K} w_{i,k}(\bar{\mathbf{x}})}. \tag{16}$$

The corresponding surface point is $\mathbf{p}_i + \mathbf{r}_i(\bar{x}^1\mathbf{t}_i + \bar{x}^2\mathbf{b}_i) + h_i(\bar{\mathbf{x}})\mathbf{n}_i$, restricted to the bounded cell.

**Directional appearance.** Each detail site stores $L$ appearance directions $\boldsymbol{\alpha}_{i,k}^{(l)} \in \mathbb{R}^3$ and corresponding colors $\mathbf{c}_{i,k}^{(l)} \in \mathbb{R}^3$. For a unit viewing direction $\mathbf{d}$, its color is

$$\begin{aligned} \mathbf{c}_{i,k}(\mathbf{d}) &= \frac{\sum_{l=1}^{L} w_{i,k}^{(l)}(\mathbf{d})\, \mathbf{c}_{i,k}^{(l)}}{\sum_{l=1}^{L} w_{i,k}^{(l)}(\mathbf{d})}, \\ w_{i,k}^{(l)}(\mathbf{d}) &= \exp\Big(-\|\mathbf{d} - \hat{\boldsymbol{\alpha}}_{i,k}^{(l)}\|_2\Big)\,, \end{aligned} \tag{17}$$

where $\hat{\boldsymbol{\alpha}}_{i,k}^{(l)} = \boldsymbol{\alpha}_{i,k}^{(l)} / \|\boldsymbol{\alpha}_{i,k}^{(l)}\|_2$. We then blend the detail-site colors using the spatial weights from Eq. 15:

$$\mathbf{c}(\bar{\mathbf{x}}, \mathbf{d}) = \frac{\sum_{k=1}^{K} w_{i,k}(\bar{\mathbf{x}})\, \mathbf{c}_{i,k}(\mathbf{d})}{\sum_{k=1}^{K} w_{i,k}(\bar{\mathbf{x}})}. \tag{18}$$

**Rest state and fixed attributes.** For completeness, the reconstructed rest scene contains

$$\mathcal{S}_0 = \left\{\mathbf{p}_i^0, \mathbf{r}_i^0, \mathbf{q}_i^0, \mathbf{s}_i, d_i, \boldsymbol{\sigma}_i, \boldsymbol{\alpha}_i^0, \mathbf{c}_i\right\}_{i=1}^{N}, \tag{19}$$

where $\mathbf{c}_i$ collects the colors $\{\mathbf{c}_{i,k}^{(l)}\}_{k,l}$. Simulation changes the positions, radii, frames, and appearance directions. Detail-site coordinates, normal displacements, densities, and color coefficients remain fixed. In particular, Eq. 2 scales the in-plane offsets through $\mathbf{r}_i$, while the normal displacements $d_{i,k}$ retain their original magnitudes.

### A.3 Material Point Method Simulation Loop

In this section, we provide additional background details on MPM simulation loop. We first define Lamé coefficients as:

$$\mu = \frac{E}{2(1+\nu)}, \qquad \lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}, \tag{20}$$

**Particle-to-grid (P2G) transfer.** Mass and momentum are accumulated onto the grid with the affine particle-in-cell (APIC) scheme (Jiang et al., 2015), augmenting each particle's velocity with a local affine velocity term $\mathbf{C}_p^t$:

$$m_i^t = \sum_p w_{ip}^t \, m_p, \tag{21}$$

$$(m\mathbf{v})_i^t = \sum_p w_{ip}^t \, m_p \left[\mathbf{v}_p^t + \mathbf{C}_p^t \left(\mathbf{x}_i^t - \mathbf{x}_p^t\right)\right], \tag{22}$$

where $w_{ip}^t$ is the B-spline weight coupling particle $p$ to grid node $i$.

**Grid update.** Each grid node velocity is integrated forward under the net nodal force $\mathbf{f}_i$, combining internal and external forces:

$$\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + \frac{\Delta t}{m_i^t} \, \mathbf{f}_i\big(\mathbf{x}_i^t; \boldsymbol{\theta}_p\big) \, . \tag{23}$$

The force follows from a hyperelastic energy $\Psi(\mathbf{F})$, and $\theta_p$ gathers the governing material properties, Young's modulus $E$, and Poisson's ratio $\nu$.

**Grid-to-particle (G2P) transfer.** The updated nodal velocities are interpolated back to the particles, whose positions are then advanced, the local affine velocity term is also updated correspondingly:

$$\mathbf{v}_p^{t+1} = \sum_i w_{ip}^t \, \mathbf{v}_i^{t+1}, \qquad \mathbf{x}_p^{t+1} = \mathbf{x}_p^t + \Delta t \, \mathbf{v}_p^{t+1}, \qquad \mathbf{C}_p^{t+1} = \frac{4}{(\Delta x)^2} \sum_i w_{ip}^t \mathbf{v}_i^{t+1} (\mathbf{x}_i - \mathbf{x}_p^t)^T \tag{24}$$

**Deformation gradient update.** Each particle's deformation gradient is updated from the velocity gradient sampled off the grid:

$$\mathbf{F}_p^{t+1} = \left[\mathbf{I} + \Delta t \sum_i \mathbf{v}_i^{t+1} \left(\nabla w_{ip}^t\right)^\top\right] \mathbf{F}_p^t. \tag{25}$$

### A.4 Additional Implementation Details

**Material Field Optimization.** Following PhysDreamer (Zhang et al., 2024), we represent both velocity field and material field using smooth spatial fields: a triplane (Chan et al., 2022), queried at each primitive's rest position $\mathbf{p}_i^0$. For triplane representation, each field uses three $24 \times 24$ feature planes with 32 channels and a two-layer MLP decoder of width 64. The velocity field is zero-initialized so the object starts at rest. The material field, with weights $\phi$, outputs both components of $\boldsymbol{\theta}_p = (E_p, \nu_p)$. We optimize Young's in log-scale space around an initial guess $E$ and clamped to a plausible range $[1e2, 1e10]$, while Poisson's ratio is bounded to its physical interval $[0, 0.5]$. We optimize with AdamW (weight decay $10^{-4}$) under a linear warmup over the first 10% of iterations

Table 3: **List of constitutive models used for simulation.**

| Scene | Figure | Constitutive Model |
|---|---|---|
| Bonsai | Fig. 1 | Fixed corotated |
| Bread roll | Fig. 2 | Fixed corotated |
| Carnation | Fig. 4 | Fixed corotated |
| GSO benchmark | Fig. 6 | Fixed corotated |
| Bread twist | Fig. 7 | Fixed corotated |
| Car | Fig. 8 | Fixed corotated |
| Can | Fig. 10 | von Mises |
| Telephone | Fig. 10 | Fixed corotated |
| Microphone | Fig. 10 | Fixed corotated |
| Ficus | Fig. 10 | Fixed corotated |
| Pillow | Fig. 10 | Fixed corotated |
| Wolf | Fig. 10 | Drucker–Prager |
| Paper Plane | Fig. 8 | Fixed corotated |

Table 4: Ablation studies on discounting factor $\beta$ with mIoU / mAcc per scene.

| $\beta$ | garden | bonsai | room | counter | kitchen | average |
|---|---|---|---|---|---|---|
| 0.1 | 0.912 / 0.948 | **0.837** / 0.881 | 0.604 / 0.953 | 0.758 / 0.912 | 0.855 / 0.947 | 0.793 / **0.928** |
| 0.2 | 0.916 / 0.948 | 0.833 / 0.877 | 0.608 / **0.955** | 0.758 / 0.912 | 0.855 / 0.947 | 0.794 / **0.928** |
| 0.4 | **0.922** / 0.948 | 0.826 / 0.868 | 0.615 / 0.953 | 0.759 / 0.912 | 0.858 / 0.947 | **0.796** / 0.926 |
| 0.5 | 0.920 / 0.945 | 0.822 / 0.863 | 0.619 / 0.953 | **0.759** / 0.912 | 0.859 / 0.947 | **0.796** / 0.924 |
| 0.6 | 0.920 / 0.944 | 0.819 / 0.860 | 0.622 / 0.952 | 0.759 / 0.912 | 0.859 / 0.947 | **0.796** / 0.923 |
| 0.8 | 0.917 / 0.940 | 0.814 / 0.853 | 0.625 / 0.948 | 0.759 / 0.912 | 0.859 / 0.947 | 0.795 / 0.920 |
| 1.0 | 0.916 / 0.938 | 0.810 / 0.848 | **0.630** / 0.939 | 0.759 / 0.912 | **0.861** / 0.947 | 0.795 / 0.917 |

followed by linear decay to zero, clipping the gradient norm to 1.0, with the photometric loss of Eq. 11 at $\lambda = 0.2$. Stage 1 fits the velocity field for 30 iterations on the first three frames at learning rate $10^{-2}$. Stage 2 freezes the velocity and fits the material field on the full video for 20 iterations at $5 \times 10^{-3}$, initialized at $\bar{E} = 10^7$ Pa.

**Simulation Details.** We provide a list of constitutive models that we used for simulation in each scene in Tab.3

### A.5 Additional Results

**Additional Physics Simulation Results** As shown in Fig.10, we provide more simulation results spanning a range of objects, materials, and interactions. Our results are best viewed on our project website: `https://iclr14573.github.io/`.

**3D Semantic Segmentation** To evaluate the performance of our proposed optimization-free dynamic primitive selection, we follow Semantic Foam(Sharafeldin et al., 2026) and evaluate on the 3D segmentation task using five Mip-NeRF 360 (Barron et al., 2022) for fair comparison with its released object masks, reporting mean Intersection over Union (mIoU) and mean Accuracy (mAcc). Tab. 5 reports quantitative results. Without any optimization, our primitive selection is competitive with methods that train a dedicated semantic field. We additionally show objects extracted by our method alongside those of Semantic Foam's in Fig.11.

**Ablation Studies on discounting factor.** Tab.4 varies the discounting factor $\beta$ from 0.1 to 1.0. We use $\beta = 0.5$ for all scenes.

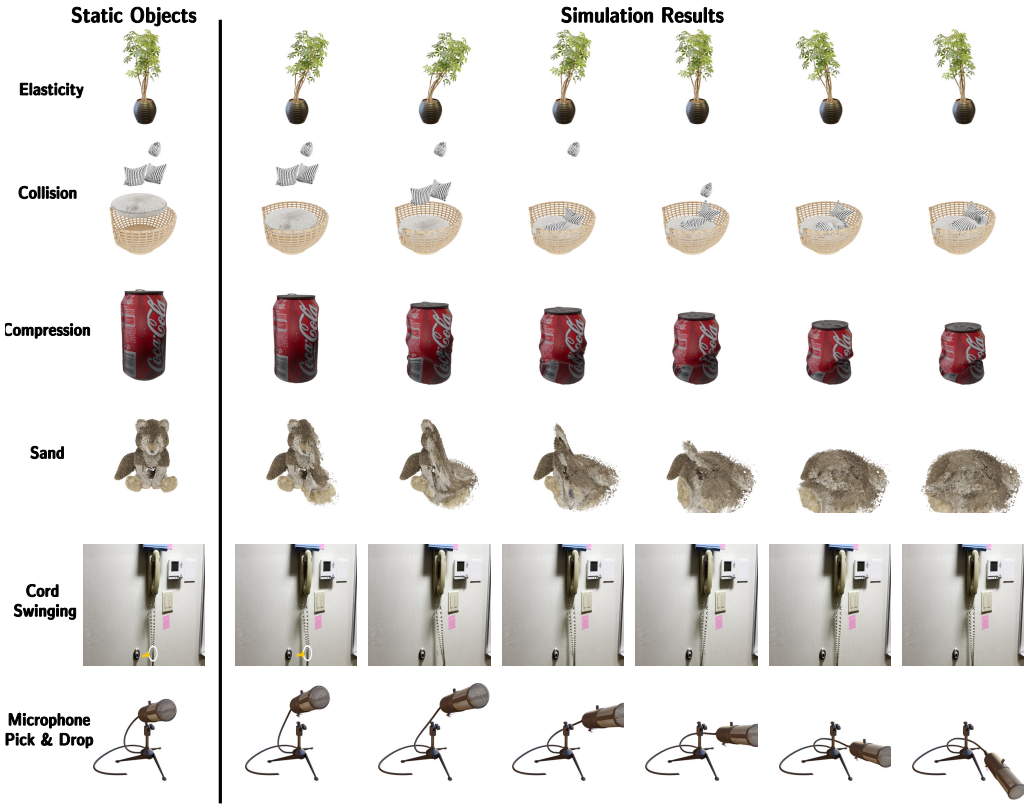


Figure 10: **Diverse Material Behaviors.** We present more simulation results on different material settings.

Table 5: Per-scene segmentation result on Mip-NeRF 360. Baseline numbers are taken from the SemanticFoam paper. All baselines optimize a per-scene semantic field; ours requires **no optimization**.

| Method | Opt.-free | Garden mIoU↑ / mAcc↑ | Bonsai mIoU↑ / mAcc↑ | Room mIoU↑ / mAcc↑ | Counter mIoU↑ / mAcc↑ | Kitchen mIoU↑ / mAcc↑ | Average mIoU↑ / mAcc↑ |
|---|---|---|---|---|---|---|---|
| LabelGS | ✗ | 0.79 / 0.95 | 0.70 / 0.92 | **0.64** / 0.93 | 0.55 / **0.93** | 0.82 / **0.96** | 0.70 / **0.94** |
| Gaussian Grouping | ✗ | 0.88 / 0.92 | 0.75 / 0.82 | 0.55 / 0.78 | 0.74 / 0.92 | 0.83 / 0.92 | 0.75 / 0.87 |
| SemanticFoam | ✗ | **0.94** / **0.96** | **0.90** / **0.94** | 0.63 / **0.95** | 0.75 / 0.91 | **0.90** / 0.94 | **0.82** / **0.94** |
| Ours | ✓ | 0.92 / 0.95 | 0.82 / 0.86 | 0.62 / **0.95** | **0.76** / 0.91 | 0.86 / 0.95 | 0.80 / 0.92 |

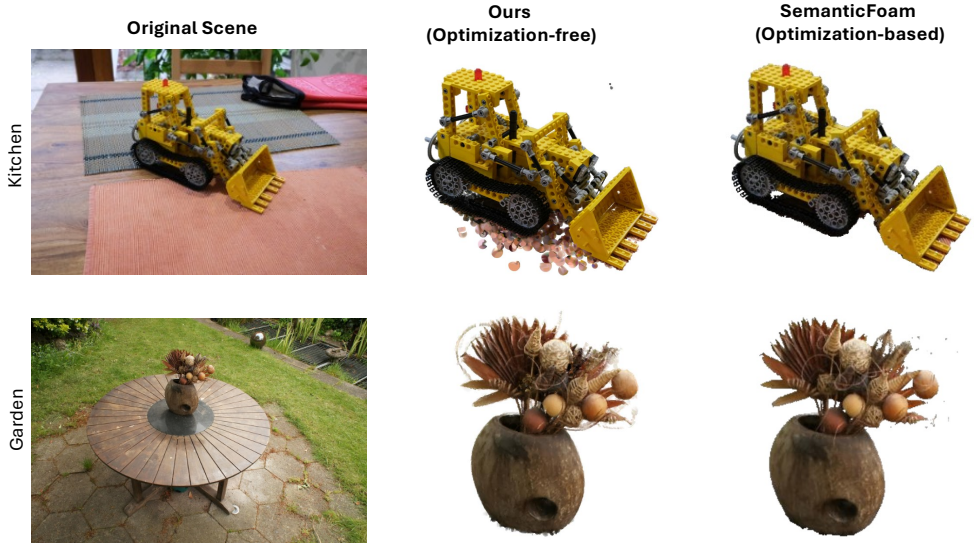


Figure 11: We provide qualitative results compared with SemanticFoam on Mip-NeRF 360 scenes

### A.6 Assets Info

We provide more information about the assets that we use in the multi-object simulation. Other simulation scenes or objects are taken from PhysGaussian Xie et al. (2023) or PhysDreamer Zhang et al. (2024).

**Car**: https://superspl.at/scene/bdb92f05

**Paper Plane**: https://sketchfab.com/3d-models/paper-plane-6e9201cd4b614d879741ea79524c57c6

**Can**: https://sketchfab.com/3d-models/french-coke-can-606cf0ae5cb44fa384bf2586982f7163