Seeing the Unseen Physics from a Single Video

Jun Dai
University of Rochester
Rochester, New York, USA
jdai20@ur.rochester.edu

Sheng Zhao
University of Rochester
Rochester, New York, USA
szhao32@ur.rochester.edu

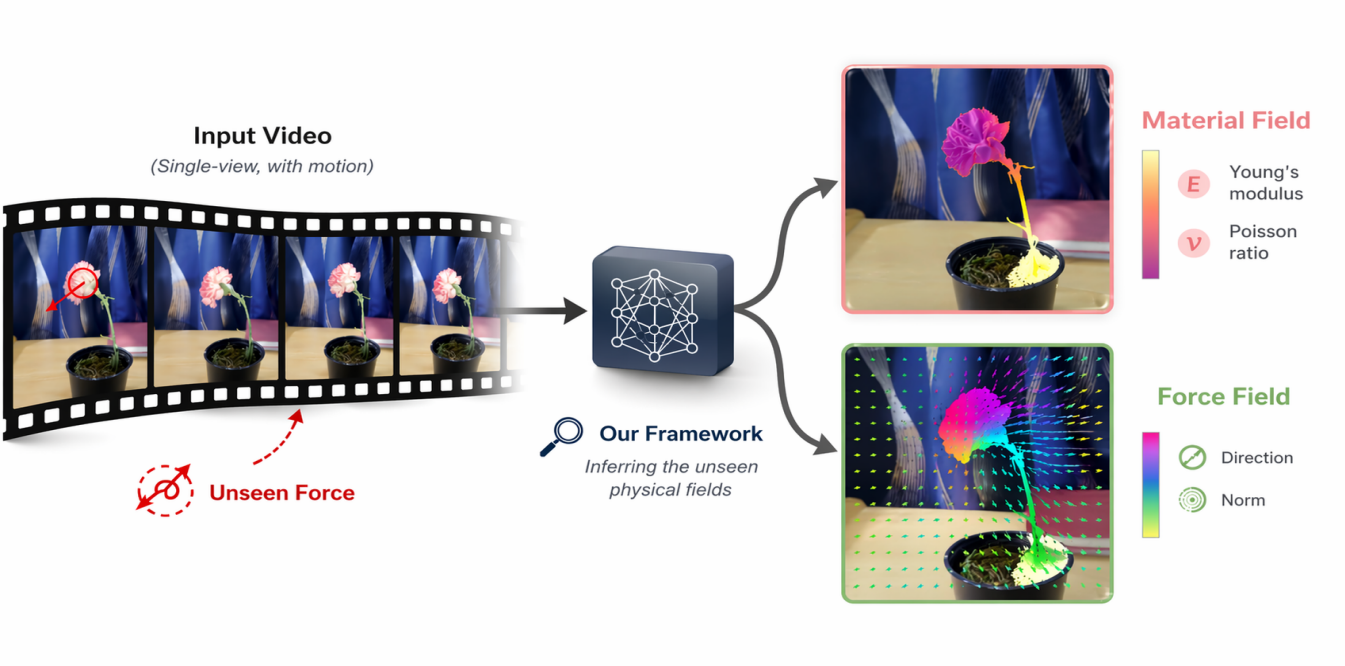


Figure 1: Seeing the unseen physics from a single video. Given an input video depicting an object in motion (left), our framework jointly recovers the underlying physical fields that govern the observed dynamics (right): a *material field* encoding spatially varying Young's modulus, Poisson's ratio, and density, and a *force field* encoding the direction and magnitude of external forces acting on the object. The color-coded visualizations on the object surface illustrate the spatial distribution of the recovered physical quantities.

Abstract

Endowing machines with the ability to infer the physical causes behind visual observations is a long-standing goal in computer vision and graphics. In this paper, we address the problem of jointly recovering invisible external forces and internal material properties from monocular videos. We present DiffPhys, a unified end-to-end optimization framework that reconstructs object geometry, spatially varying material fields, and spatio-temporally varying force fields within a single differentiable pipeline. By coupling visual reconstruction with a Material Point Method simulator, DiffPhys optimizes object motion under physically plausible dynamics while explicitly modeling the interactions among geometry, materials, and forces. Extensive experiments on the PhysDreamer benchmark show that DiffPhys recovers high-fidelity physical fields and significantly outperforms prior methods that estimate materials or forces in isolation. The project webpage is available at: https://daijun10086.github.io/PhysFields/.

1 Introduction

Videos reveal how objects move, yet the physical causes behind that motion remain hidden: a flower bending in the wind, a branch swaying under an unseen load, or a cloth deforming after contact all expose vivid image-space motion while concealing two coupled latent factors—the external forces that drive it and the internal material properties that shape its response. Recovering these latent fields from ordinary monocular videos would enable physically grounded scene understanding, simulation-based editing, robotic interaction, and force or material retargeting.

Recovering the two fields jointly from a single video is fundamentally challenging. Both are high-dimensional and spatially varying, and observed only indirectly through motion: external forces vary across space and time, real materials are non-uniform in stiffness, density, and deformation behavior, and the two interact through the object's dynamics rather than as independent visual cues. The monocular setting amplifies the difficulty, providing only partial observations of 3D shape and motion. Recovering the physical

Machine Vision, University of Rochester
2026.

causes behind visible motion therefore requires a model that reasons jointly over time-varying forces, spatially varying materials, and their coupled influence on dynamics.

Existing approaches address pieces of this problem but none of them jointly recover geometry, force, and material from monocular video. Dynamic NeRF [6, 19] and Gaussian Splatting [11, 15, 25, 27] representations capture time-varying geometry and appearance, but remain descriptive and reveal no underlying physics. Physics-integrated representations and differentiable simulators [13, 26] enable plausible animation, yet require user-supplied material parameters or external inputs. PhysDreamer [28] and Physics3D [14] infer material properties via video generation priors but assume the external force is known; conversely, force-oriented methods [1, 12] recover external fields while relying on fixed or coarsely initialized materials. Existing work therefore models motion without latent physics, recovers material without forces, or recovers forces without material—never the full pair.

Motivated by these limitations, we present *DiffPhys*, a unified end-to-end framework that jointly recovers object geometry, spatially varying material fields, and spatio-temporally varying force fields from monocular video. DiffPhys represents the object with 3D Gaussian Splatting for differentiable rendering and couples it with a differentiable Material Point Method (MPM) simulator [10] for physically grounded dynamics. The material field is parameterized over space, the force field over space and time, and both act on the simulated object. By rendering the simulated state back to image space, DiffPhys supervises the latent fields directly from video while keeping the entire path through geometry, simulation, and rendering differentiable.

Beyond this unified formulation, we introduce a progressive optimization strategy that first estimates the force field under fixed material assumptions and then jointly refines both fields, alleviating force–material ambiguity and avoiding incorrect but visually plausible solutions.

Our contributions are summarized as follows:

- We formulate monocular inverse physics as the joint recovery of geometry, materials, and time-varying forces from a single video.
- We introduce *DiffPhys*, an end-to-end differentiable framework that couples 3D Gaussian Splatting with MPM simulation under direct image-space supervision.
- We design a progressive force-then-joint optimization scheme that alleviates the inherent ambiguity between the two fields.
- We validate our framework on the PhysDreamer [28] benchmark, demonstrating that joint material–force recovery yields markedly more faithful video reconstructions than prior methods that infer either field alone.

2 Related Work

2.1 4D Dynamic Scene Reconstruction

Reconstructing 3D scenes from images has been broadly explored through implicit Neural Radiance Fields (NeRF) [17] and explicit 3D Gaussian Splatting (3DGS) [11]; the latter, with its real-time differentiable rasterization and explicit point-based structure, has emerged as a popular foundation for downstream tasks. A natural extension is to model temporally evolving scenes. NeRF-based approaches model dynamics by conditioning the radiance field on a temporal input [6] or by learning a deformation field over a canonical template [19]. Gaussian-based methods follow analogous strategies, including per-Gaussian trajectory tracking [15], canonical-space deformation networks, and 4D Gaussian primitives [25, 27]. These methods produce high-fidelity free-viewpoint renderings of *observed* object dynamics and have substantially advanced 4D scene reconstruction. However, they capture only the geometric and appearance changes of the scene over time, treating motion as a purely descriptive phenomenon: trajectories are fitted to the data without any underlying physical model. As a result, the recovered representation cannot be re-simulated under new conditions, cannot answer counterfactual questions such as *what would the scene look like under a stronger wind or with a stiffer material?*, and offers no guarantee of physical plausibility once driven beyond the captured time window. We adopt 3DGS as the underlying scene representation in this work but, in contrast to these methods, recover the latent *physical fields*—force and material—that generate the observed motion under principled simulation, yielding a representation that is both reconstructive and predictive.

2.2 Physical Simulation

The Material Point Method (MPM) [10, 22, 23] is a hybrid Eulerian–Lagrangian continuum simulator that handles elastic, plastic, granular, and fluid materials within a single unified formulation, making it especially suitable for the diverse object behaviors observed in everyday video. Recent differentiable physics frameworks—DiffTaichi [8], ChainQueen [9], and NVIDIA Warp [16]—further expose analytic derivatives of the simulated trajectory with respect to material and force parameters, making MPM directly usable as a layer in gradient-based learning pipelines. Building on this foundation, a line of work couples differentiable MPM with 3D Gaussian Splatting by treating each Gaussian directly as a simulation particle and rendering the deformed Gaussians as the animated output [26]. While these systems demonstrate that physics and rendering can be unified in a single forward pass, they remain *forward*-only: material properties and external forces must be user-prescribed, and no physical quantities are recovered from observation. We adopt differentiable MPM (via NVIDIA Warp) as our simulation backbone, but use it in a closed loop with video supervision to *infer* the underlying physics rather than animate prescribed physics.

2.3 Inverse Physics from Video

Building on the differentiable simulators of Section 2.2, a line of *inverse physics from video* work closes the loop between simulation and observation: forward simulation produces a rendered trajectory, and backward gradient flow optimizes physical parameters so that the rendered output matches the input video. Existing efforts can be grouped by the type of physics they recover. The first line focuses on the *material field*, assuming external forces are known. PAC-NeRF [13] couples differentiable MPM with NeRF to identify material parameters from multi-view video using direct image supervision. PhysDreamer [28] and Physics3D [14] go a step further by distilling motion priors from pre-trained video diffusion models [3, 7] via Score Distillation Sampling [18, 24] to constrain the recovered materials. The second line focuses on the *external*

force field: Phy124 [1] and DiffWind [12] estimate spatio-temporal forces—e.g., wind—driving observed motion, with material parameters initialized from a vision-language model (VLM) and held fixed during optimization. In contrast, we jointly recover both fields from a single video.

3 Method

3.1 Problem Statement

Given a monocular video $\mathcal{V} = \{I_t\}_{t=1}^{T}$ of an object in motion and an initial geometric state $\mathcal{G}_0$, our goal is to recover two latent physical fields that explain the observed dynamics: a spatially varying *material field*

$$\mathbf{m}_{\theta_m} : \Omega \to \mathbb{R}^{d_m} \tag{1}$$

encoding local material properties (e.g., Young's modulus E), and a spatio-temporal *force field*

$$\mathbf{f}_{\theta_f} : \Omega \times \{1, \dots, T\} \to \mathbb{R}^3 \tag{2}$$

encoding external forces, where $\Omega \subset \mathbb{R}^3$ is the object domain and (θ_m, θ_f) are learnable field parameters.

Our key objective is to jointly optimize (θ_m, θ_f) so that the dynamics simulated under the two fields, when rendered, match the observed video. To this end, we couple a differentiable MPM simulator $\mathcal{S}$ with a differentiable renderer $\mathcal{R}$: starting from the initial state $\mathcal{G}_0$, the simulator $\mathcal{S}$ evolves the object under the two fields, and the renderer $\mathcal{R}$ projects each simulated state back to image space through a fixed camera C_0. The composition produces the predicted frame at time t,

$$\hat{I}_t \;=\; \mathcal{R}\Big(\mathcal{S}(\mathcal{G}_0;\, \mathbf{m}_{\theta_m},\, \mathbf{f}_{\theta_f},\, t);\, C_0\Big), \tag{3}$$

and we optimize (θ_m, θ_f) such that the rendered video $\hat{\mathcal{V}} = \{\hat{I}_t\}_{t=1}^{T}$ matches the input $\mathcal{V}$ in image space.

3.2 Data Preprocessing

Before optimization, we run two data-driven preprocessing steps that produce the static scene representation $\mathcal{G}_0$ and a coarse initialization of the material field. Both steps are detached from the main optimization loop and incur no gradient cost during training.

Static Gaussian Splatting initialization. We extract the first frame I_0 from the input video and segment the foreground object with an Otsu-based mask. A monocular depth estimator (MiDaS [20]) is then run on the masked frame, and the foreground pixels are back-projected through the camera C to produce an initial point cloud whose color is read directly from I_0. We optimize a 3D Gaussian Splatting model [11] on this point cloud against I_0 for several thousand iterations with densification and pruning, yielding the static representation

$$\mathcal{G}_0 = \{(\boldsymbol{\mu}_i, \Sigma_i, \alpha_i, \mathbf{c}_i)\}_{i=1}^{N}, \tag{4}$$

which encodes both the geometry and the appearance of the object under the fixed camera and serves as the starting state of every subsequent simulation.

VLM-based material initialization. A reasonable early-stage initialization of the material field is critical for end-to-end optimization to converge: starting from homogeneous defaults or extreme stiffness values destabilizes training, with particles either overdeforming under any reasonable force or refusing to move at all. A VLM provides a coarse but physically reasonable prior, and is further naturally part-aware — an everyday object usually consists of regions with very different physical materials (e.g., a flower has stiff stems and soft petals), and the VLM can name these parts and assign each a distinct material, giving the subsequent optimization a meaningful spatial pattern to refine rather than a uniform default. Concretely, we feed I_0 to the VLM together with SAM-generated part masks; the VLM names each part, infers its physical material, and we map the result to per-part values of Young's modulus E, Poisson's ratio ν, and density ρ via a fixed material database. These per-part values are propagated to MPM particles inside each segmented region (with nearest-neighbor fill for unassigned particles), giving per-particle targets $(E_i^{\text{vlm}}, \nu_i^{\text{vlm}}, \rho_i^{\text{vlm}})$. We then pretrain $\mathbf{m}_{\theta_m}$ by MSE regression onto these targets so that the subsequent end-to-end optimization starts from a physically meaningful point. Density ρ^{vlm} is held fixed throughout training; the (E, ν) branches are unfrozen during the joint stage in Section 3.5.

3.3 Physical Simulation

Continuum mechanics. The object dynamics we seek to recover are governed by continuum mechanics, which describes the motion through a time-dependent deformation map $\mathbf{x} = \boldsymbol{\phi}(\mathbf{X}, t)$ between the rest configuration $\mathbf{X}$ and the deformed configuration $\mathbf{x}$, with the deformation gradient $\mathbf{F} = \nabla_{\mathbf{X}}\boldsymbol{\phi}$ encoding local stretch, rotation, and shear. The evolution of $\boldsymbol{\phi}$ is governed by the conservation of mass and momentum,

$$\frac{D\rho}{Dt} + \rho\, \nabla\cdot \mathbf{v} = 0, \qquad \rho\, \dot{\mathbf{v}} \;=\; \nabla\cdot \boldsymbol{\sigma} + \mathbf{f}^{\text{ext}}, \tag{5}$$

where ρ is the mass density, $\mathbf{v}$ is the velocity field, $\boldsymbol{\sigma}$ is the Cauchy stress derived from a hyperelastic energy density $\Psi(\mathbf{F}; E, \nu)$ parameterized by the local material constants, and $\mathbf{f}^{\text{ext}}$ is the external body force. Recovering motion under our two unknown fields $\mathbf{m}_{\theta_m}$ and $\mathbf{f}_{\theta_f}$ thus reduces to numerically integrating this coupled system forward in time.

Material Point Method. We solve the continuum equations with the Material Point Method (MPM) [10, 23], which combines the strengths of a Lagrangian and an Eulerian discretization. Mass conservation is naturally maintained on a set of moving Lagrangian particles carrying $(\mathbf{x}_p, \mathbf{v}_p, \mathbf{F}_p)$ as they travel with the matter, while momentum conservation is solved on a static Eulerian background grid where spatial differentiation is straightforward. Two-way transfer between the two representations is performed with B-spline kernels w_{ip}^n defined between particle p and grid node i.

A single MPM substep from t^n to t^{n+1} proceeds in three stages. In *P2G*, particle mass and momentum are scattered to the grid as $m_i^n = \sum_p m_p w_{ip}^n$ and $(m\mathbf{v})_i^n = \sum_p m_p \mathbf{v}_p^n w_{ip}^n$. The discretized momentum equation is then integrated on the grid under elastic stress and external force,

$$\frac{m_i}{\Delta t}\big(\mathbf{v}_i^{n+1} - \mathbf{v}_i^n\big) \;=\; -\sum_p V_p^0\, \boldsymbol{\sigma}_p^n\, \nabla w_{ip}^n \;+\; \mathbf{f}_i^{\text{ext}}, \tag{6}$$

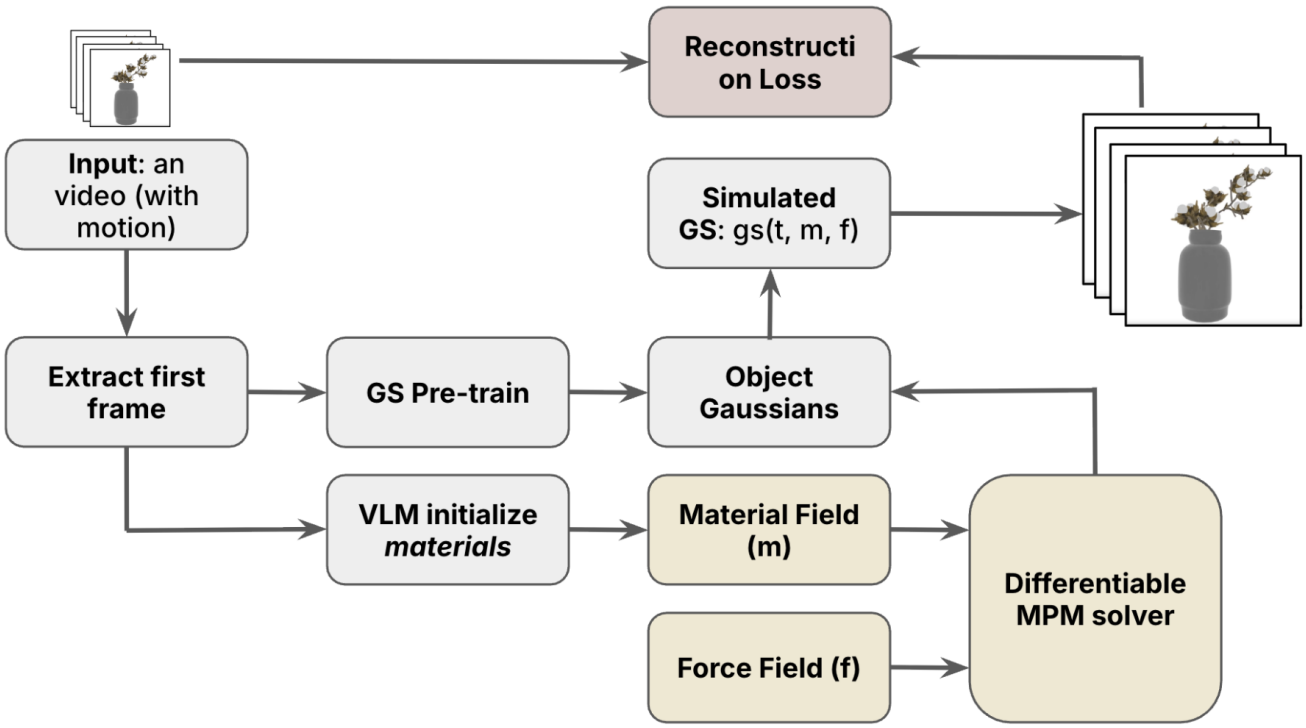


Figure 2: Pipeline of DiffPhys. Given a monocular video and the static Gaussian Splatting scene reconstructed from its first frame, we feed only the moving particles into a differentiable Material Point Method (MPM) simulator. Two latent fields drive the simulation: a per-particle (Lagrangian) *material field* that supplies Young's modulus and Poisson's ratio at each particle's current position, and a per-grid (Eulerian) *force field* that injects an external force at every voxel of the MPM background grid. The deformed Gaussians are rendered through a differentiable rasterizer under the fixed input camera, and the rendered video is compared against the input through a foreground-masked reconstruction loss. The whole pipeline is end-to-end differentiable, enabling joint recovery of both fields directly from video supervision.

where V_p^0 is the rest volume of particle p, $\boldsymbol{\sigma}_p^n$ is the per-particle Cauchy stress evaluated from $\mathbf{F}_p^n$ and the queried material constants, and $\mathbf{f}_i^{\text{ext}}$ is the external force at grid node i, which we will read off from the force field below. Finally, in *G2P* the updated grid velocity is gathered back to the particles to advance position and deformation:

$$\mathbf{v}_p^{n+1} = \sum_i w_{ip}^n \mathbf{v}_i^{n+1}, \quad \mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \Delta t\, \mathbf{v}_p^{n+1}, \quad \mathbf{F}_p^{n+1} = \left(\mathbf{I} + \Delta t\, \nabla \mathbf{v}_p^{n+1}\right)\mathbf{F}_p^n. \tag{7}$$

We chain K such substeps per video-frame interval to satisfy the explicit-MPM CFL stability condition; the resulting particle deformation then drives the associated Gaussians, which are rendered to image space through $\mathcal{R}$.

3.4 Physical Field Representation

The two unknown fields $\mathbf{m}_{\theta_m}$ and $\mathbf{f}_{\theta_f}$ enter the MPM solver above at fundamentally different sites: the material field supplies the per-particle stress, while the force field provides the per-grid external force. We parameterize each at the site where it is consumed.

Material field — per-particle (Lagrangian) query. Material constants are intrinsic properties of matter: Young's modulus and Poisson's ratio describe how a small piece of material responds to local deformation, regardless of where it happens to be at any instant. We therefore parameterize $\mathbf{m}_{\theta_m}$ at the Lagrangian level: each MPM particle queries the field at its own current position to read out its (E_p, ν_p), which then feed directly into the per-particle stress $\boldsymbol{\sigma}_p^n$ in Eq. (6). We implement this query with a TriPlane [5] feature backbone Π_{θ_m} shared by two independent MLP heads, one per material attribute:

$$E_p = h_E\big(\Pi_{\theta_m}(\mathbf{x}_p)\big), \qquad \nu_p = h_\nu\big(\Pi_{\theta_m}(\mathbf{x}_p)\big), \tag{8}$$

where each head is bounded to a physically meaningful range. The field is initialized from the VLM prior of Section 3.2.

Force field — per-grid (Eulerian) query. External forces are properties of space and time, not of any specific particle: at any given location and time, the same force acts on whatever matter happens to be there, regardless of which particle currently occupies that point. Anchoring the force on particles would tie it to the current Lagrangian sample set; we instead parameterize $\mathbf{f}_{\theta_f}$ directly on the MPM background grid. Each voxel carries its own learnable feature vector, and a small MLP head decodes the queried feature into the

Table 1: Reconstruction quality on the PhysDreamer [28] benchmark. Per-frame foreground-masked PSNR / SSIM / LPIPS between the rendered and input video, averaged across the four scenes. Higher PSNR / SSIM and lower LPIPS are better. Best values are bolded.

Method	PSNR↑	SSIM↑	LPIPS↓
PhysDreamer [28]	13.89	0.550	0.498
Physics3D [14]	14.72	0.590	0.452
PhysGaussian [26]	13.86	0.570	0.471
Ours	**24.92**	**0.750**	**0.241**

Table 2: Joint vs. single-field optimization on the PhysDreamer [28] benchmark. Joint material–force optimization (our default) substantially outperforms either single-field variant, confirming that the two fields are complementary and must be recovered together.

Variant	PSNR↑	SSIM↑	LPIPS↓
Material-only	13.42	0.577	0.438
Force-only	23.01	0.691	0.289
Joint (Ours)	**24.92**	**0.750**	**0.241**

3D force entering the simulation:

$$\mathbf{f}_i^{\text{ext}} \;=\; \mathbf{f}_{\theta_f}(\mathbf{x}_i, t) \;=\; h_{\theta_f}(\mathbf{Z}_t[i]), \tag{9}$$

where $\mathbf{Z}_t$ is a per-frame learnable feature grid (one feature vector per voxel, indexed by the same lattice MPM uses for momentum exchange), and h_{θ_f} is an MLP mapping the queried feature to a 3-vector force. This $\mathbf{f}_i^{\text{ext}}$ is precisely the external force entering the grid momentum update in Eq. (6), so the force enters the system as a genuine Eulerian field, decoupled from the current particle distribution.

3.5 Training

Training objective. We optimize (θ_m, θ_f) with a minimal composite objective: an image-space reconstruction loss together with a light smoothness regularizer on the force field,

$$\mathcal{L} \;=\; \mathcal{L}_{\text{recon}} \;+\; \lambda_{\text{reg}}\, \mathcal{L}_{\text{reg}}. \tag{10}$$

$\mathcal{L}_{\text{recon}}$ combines a per-frame L2 loss and an SSIM loss between the rendered video $\hat{\mathcal{V}}$ and the input $\mathcal{V}$, both evaluated only inside the foreground mask of the moving object since only the foreground deforms while the background is held fixed. $\mathcal{L}_{\text{reg}}$ imposes spatio-temporal smoothness on the force field $\mathbf{f}_{\theta_f}$, suppressing high-frequency noise that the inverse problem cannot constrain from observation alone. This minimal objective is sufficient on its own: direct video reconstruction together with a light force-field smoothness term is already enough to disentangle and recover both fields, without the generative video priors used by prior material-recovery work [14, 28].

In practice, we follow a simple two-stage schedule for stability: a short warm-up that updates only θ_f with $\mathbf{m}_{\theta_m}$ frozen at its VLM initialization, followed by joint optimization of both fields under the same objective.

Table 3: KNN downsample ratio ablation. Reconstruction quality (PSNR and SSIM, computed on the full rendered image and on the foreground-masked region) versus particle count, averaged over a 5-frame validation run on two PhysDreamer [28] scenes. Best per scene is bolded.

Scene	Ratio	# Particles	Full PSNR ↑	Masked PSNR ↑	Masked SSIM ↑
Carnations	0.01	1,335	**35.02**	**25.30**	**0.8066**
	0.02	2,671	34.81	24.62	0.7727
Hat	0.01	2,979	**40.69**	**37.68**	**0.9850**
	0.02	5,958	40.61	37.49	0.9845

Table 4: MPM grid resolution ablation on the alocasia scene. Reconstruction quality and per-iteration training cost, averaged over a 5-frame validation run. Best per metric is bolded.

Grid	Full PSNR ↑	Masked PSNR ↑	Masked SSIM ↑	Time / iter (s) ↓
64^3	**39.98**	**37.36**	**0.9846**	**1.11**
128^3	39.96	37.26	0.9841	2.35

4 Experiments

In this section, we first describe the implementation details of our framework (Sec. 4.1), then evaluate the quality of the recovered physical fields against the input video on the PhysDreamer [28] benchmark (Sec. 4.2), and finally validate key design choices through ablation studies (Sec. 4.3).

4.1 Implementation Details

For real-world captured data, we first run COLMAP [21] to recover camera poses and a sparse point cloud, then train a static 3D Gaussian Splatting [11] model on the multiview images. To isolate the deforming part of the scene, we compute optical flow between adjacent frames and threshold its magnitude to obtain a foreground mask of the moving region; the corresponding Gaussians form the simulated object, while the static background is held fixed throughout training. Per-part physical property initialization is obtained by prompting GPT-5.5 with the frames and a fixed material database, as described in Section 3.2.

In our experiments, we discretize the simulation space into a 64^3 background grid and feed only the moving-region particles to the MPM solver, which is built on NVIDIA Warp [16]. All experiments are run on a single NVIDIA GeForce RTX 4090 GPU. Additional hyperparameters — TriPlane and feature-grid resolutions, MLP architecture, learning rates, and per-stage iteration counts — are deferred to the appendix.

4.2 Evaluation on Physical Field Recovery

Datasets. We evaluate on the PhysDreamer [28] benchmark, which provides four monocular video scenes of soft objects (plants and cloth) deforming under external interaction. The PhysDreamer scenes cover the regimes our framework targets—spatially varying material distributions and externally driven motion—and span enough geometric diversity (a flowering carnation, broad alocasia leaves, a knit hat, and a coiled telephone cord) to stress both the material and force branches of our pipeline.

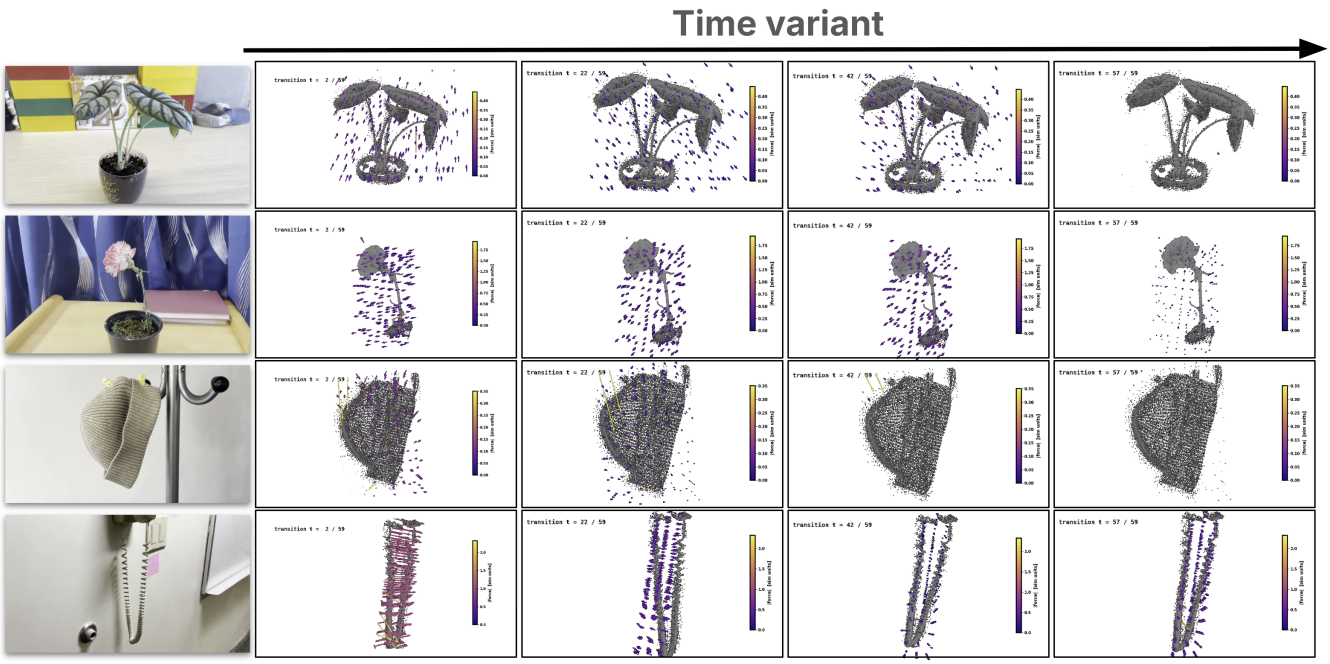


Figure 3: Recovered spatio-temporal force fields across four PhysDreamer [28] scenes. Each row corresponds to one scene (top to bottom: alocasia, carnations, hat, telephone). The leftmost column shows the input video reference frame; the four panels to the right visualize the force field recovered by our framework at four uniformly spaced time steps within the sequence, with arrows indicating the per-particle force direction and the colorbar encoding its magnitude. Across all four scenes, the recovered forces concentrate precisely on the moving parts and reverse direction in lockstep with the object's oscillation, while remaining near zero elsewhere—demonstrating that our reconstruction is both spatially localized to the deforming regions and temporally well aligned with the observed motion.

Baselines. Our closest relative is DiffWind [12], but no open-source code is available, so we cannot compare against it directly. We therefore fall back to PhysDreamer [28] and Physics3D [14]; these methods recover only the material field, whereas we recover both fields. To align them with our setting, we take their recovered material, supply the ground-truth (or otherwise specified) force at simulation time, forward-simulate, and compare the resulting video against the input. We additionally include PhysGaussian [26] as a forward-only reference, fed with ground-truth $(E, \nu, \mathbf{F})$ when available.

Metrics. Since the comparison reduces to matching the reconstructed video against the input, we use the standard image-space metrics on the PhysDreamer [28] videos: foreground-masked PSNR, SSIM, and LPIPS. Qualitative visualizations of the recovered fields and the simulated motion complement these scalar metrics.

Results. As shown in Table 1, our framework outperforms every prior method by a substantial margin—more than 10 dB in PSNR. Such a large gap is itself indirect evidence of accurate field recovery: only when both the material field and the force field are faithful to the underlying physics can a forward simulation reproduce the input video this well.

Figure 3 visualizes the recovered force field over time. Both its direction and its magnitude are well aligned with the object's motion, suggesting that the inferred forces are physically consistent with the observed dynamics. Figure 4 shows the recovered material field from multiple viewpoints, which exhibits clear spatial heterogeneity that follows the object's part structure—evidence that our framework expresses genuine spatial variation of material properties rather than collapsing to a per-scene constant.

Figure 5 compares our simulated video against the input video along a fixed horizontal slice over time. The simulated slices closely follow the ground-truth ones across all four scenes, confirming that the simulated motion stays faithful to the observed motion throughout the sequence.

4.3 Ablation Studies

We conduct a series of ablation studies to understand how individual design choices in our framework contribute to the overall reconstruction quality.

Material-only, force-only, vs. joint optimization. To isolate the contribution of joint recovery, we compare three variants: material-only, force-only, and joint optimization. Both single-field variants are systematically worse than the joint version (Table 2): the material-only run cannot compensate when the prescribed force fails to explain part of the motion, while the force-only run cannot capture spatial material variation.

KNN downsampling ratio. The particle count fed to the differentiable MPM solver is bounded by GPU memory: each particle

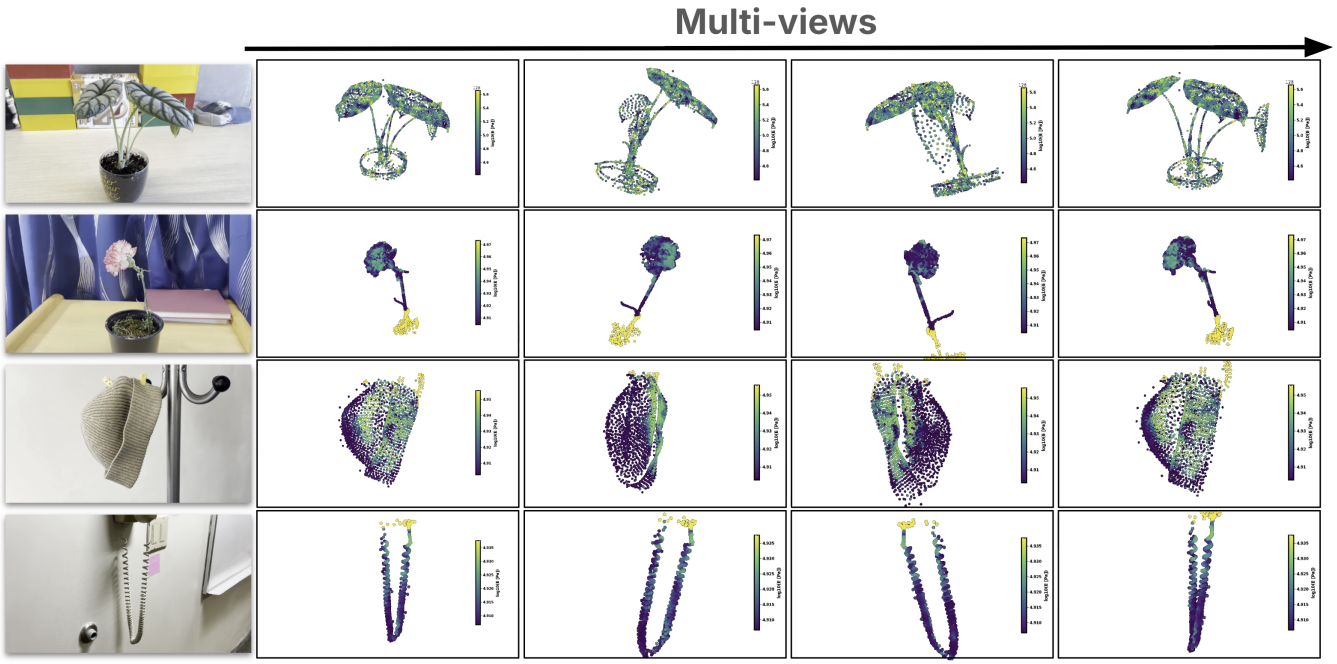


Figure 4: Recovered spatially varying material fields across four PhysDreamer [28] scenes. Each row corresponds to one scene (top to bottom: alocasia, carnations, hat, telephone). The leftmost column shows the input video reference frame; the four panels to the right show the recovered per-particle material field rendered from four different viewpoints around the object, with each particle colored by its predicted material attribute (colorbar shown per panel). The recovered field is consistent across viewpoints and exhibits clear part-aware spatial structure—e.g., distinct values on stems versus petals, or on the hat body versus its hanging accessories—confirming that our framework captures the heterogeneous material composition of each object beyond a uniform per-scene constant.

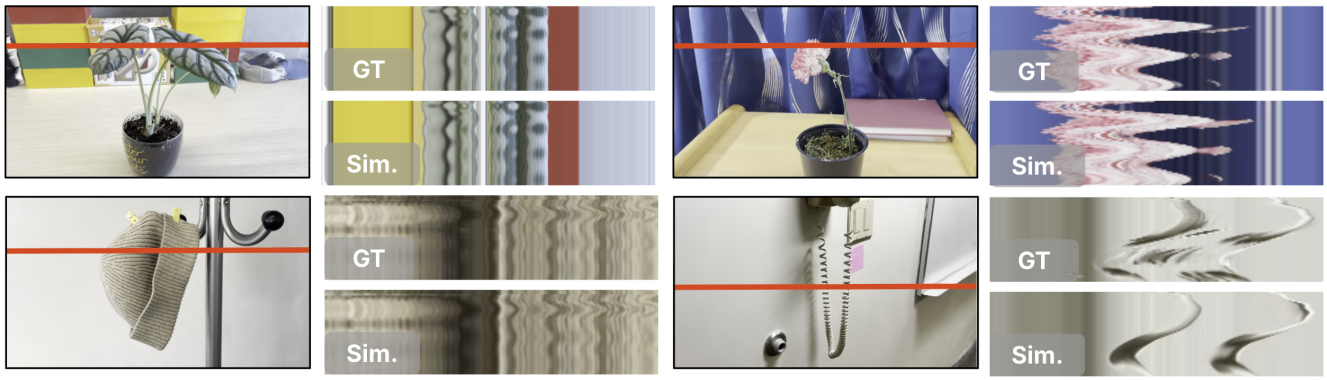


Figure 5: Spatial–temporal slice comparison on four PhysDreamer [28] scenes. In each panel, the red horizontal line on the input frame marks a fixed spatial slice; the two stacked strips on the right show how that slice evolves over time in the ground-truth video (*GT*, top) and in our simulated video (*Sim.*, bottom). The simulated strips closely follow the ground-truth ones across all four scenes, indicating that our simulated motion stays faithful to the observed motion throughout the sequence.

contributes a per-substep entry to the autograd graph, so on a 24 GB device the downsample ratio cannot grow arbitrarily. Following PhysDreamer [28], we use KNN to downsample the moving-region Gaussians, and compare two ratios within this budget, 0.01 and 0.02, on two of its scenes (Table 3). Quality is largely insensitive to this choice; we adopt 0.01, which is marginally better and uses fewer particles for lower memory and faster training.

MPM grid resolution. The MPM grid resolution trades simulation cost against fidelity. On the alocasia scene we compare the default 64^3 grid against a finer 128^3 grid. The two produce nearly

identical reconstruction quality, while the 128^3 run is more than $2\times$ slower per training iteration (Table 4); we therefore retain 64^3 as our default.

5 Conclusion

We presented *DiffPhys*, the first framework that recovers both the external force field and the spatially varying material field of an object from a single monocular video, treating the inverse problem as end-to-end optimization through a differentiable MPM simulator and a differentiable Gaussian renderer. On the PhysDreamer [28] benchmark, jointly recovering the two fields produces markedly more faithful reconstructions than prior methods that recover only material. By bringing both halves of the inverse-physics problem under a single roof, we hope this work moves video-based physical understanding closer to a tool that not only describes what is seen but also explains why the scene moves the way it does.

References

[1] 2025. Seeing the Unseen: Visual Common Sense for Semantic Placement. *arXiv preprint arXiv:2512.00762* (2025). Phy124: recovering force fields from video via 3DGS and differentiable MPM.

[2] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. 2023. Qwen-VL: A Versatile Vision-Language Model for Understanding, Localization, Text Reading, and Beyond. *arXiv preprint arXiv:2308.12966* (2023).

[3] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorber, Yam Levi, Zion English, Vikram Voleti, Adam Letts, Varun Jampani, and Robin Rombach. 2023. Stable Video Diffusion: Scaling Latent Video Diffusion Models to Large Datasets. *arXiv preprint arXiv:2311.15127* (2023).

[4] Blender Foundation. 2026. Blender. https://www.blender.org/.

[5] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. 2021. Efficient Geometry-aware 3D Generative Adversarial Networks. In *arXiv*.

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

[7] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. 2022. Video Diffusion Models. *Advances in Neural Information Processing Systems (NeurIPS)* (2022).

[8] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. 2020. DiffTaichi: Differentiable Programming for Physical Simulation. *International Conference on Learning Representations (ICLR)* (2020).

[9] Yuanming Hu, Jiancheng Liu, Andrew Spielberg, Joshua B. Tenenbaum, William T. Freeman, Jiajun Wu, Daniela Rus, and Wojciech Matusik. 2019. ChainQueen: A Real-Time Differentiable Physical Simulator for Soft Robotics. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 6265–6271.

[10] Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. 2016. The Material Point Method for Simulating Continuum Materials. In *ACM SIGGRAPH 2016 Courses*. 1–52.

[11] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 42, 4 (2023), 1–14.

[12] Yuanhang Lei, Boming Zhao, Zesong Yang, Xingxuan Li, Tao Cheng, Haocheng Peng, Ru Zhang, Yang Yang, Siyuan Huang, Yujun Shen, et al. 2026. DiffWind: Physics-Informed Differentiable Modeling of Wind-Driven Object Dynamics. *arXiv preprint arXiv:2603.09668* (2026).

[13] Xuan Li, Yi-Ling Dong, Richard Yixuan Chin, Yiheng Ni, Jingwei Sun, Kalyan Sunkavalli, Yannick Hold-Geoffroy, and Changxi Lu. 2023. PAC-NeRF: Physics Augmented Continuum Neural Radiance Fields for Geometry-Agnostic System Identification. In *International Conference on Learning Representations (ICLR)*.

[14] Fangfu Liu, Hanyang Wang, Shunyu Yao, Shengjun Zhang, Jie Zhou, and Yueqi Duan. 2024. Physics3D: Learning Physical Properties of 3D Gaussians via Video Diffusion. *arXiv preprint arXiv:2406.04338* (2024).

[15] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. 2024. Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis. *International Conference on 3D Vision (3DV)* (2024).

[16] Miles Macklin. 2022. Warp: A High-performance Python Framework for GPU Simulation and Graphics. https://github.com/NVIDIA/warp.

[17] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *European Conference on Computer Vision (ECCV)*. Springer, 405–421.

[18] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. 2023. DreamFusion: Text-to-3D using 2D Diffusion. In *International Conference on Learning Representations (ICLR)*.

[19] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. 2021. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10318–10327.

[20] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. 2020. Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-Shot Cross-Dataset Transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 3 (2020), 1623–1637.

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

[22] Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. 2013. A Material Point Method for Snow Simulation. *ACM Transactions on Graphics* 32, 4 (2013), 1–10.

[23] Deborah Sulsky, Zhen Chen, and Howard L. Schreyer. 1994. A Particle Method for History-Dependent Materials. *Computer Methods in Applied Mechanics and Engineering* 118, 1–2 (1994), 179–196.

[24] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. 2023. ProlificDreamer: High-Fidelity and Diverse Text-to-3D Generation with Variational Score Distillation. *Advances in Neural Information Processing Systems (NeurIPS)* (2023).

[25] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 2024. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 20310–20320.

[26] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. 2024. PhysGaussian: Physics-Integrated 3D Gaussians for Generative Dynamics. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[27] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. 2023. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. *arXiv preprint arXiv:2310.10642* (2023).

[28] Tianyuan Zhang, Hong-Xing Yu, Rundi Wu, Brandon Y. Feng, Changxi Zheng, Noah Snavely, Jiajun Wu, and William T. Freeman. 2024. PhysDreamer: Physics-Based Interaction with 3D Objects via Video Generation. In *European Conference on Computer Vision (ECCV)*.

A Problem Statement and Method

A.1 Problem Statement

We formulate inverse physics from monocular video as a joint field estimation problem. Given an observed video $\mathcal{V} = \{I_t\}_{t=1}^{T}$, an object domain $\Omega \subset \mathbb{R}^3$, and an initial geometric state $\mathcal{G}_0$ estimated from the first frame, our goal is to recover the latent physical quantities that best explain the observed motion. The unknowns are not trajectories themselves, but two underlying fields that generate those trajectories: a spatially varying *material field* and a spatio-temporal *force field*.

Optimization Variables. We parameterize the unknown physics by

$$\mathbf{m}_{\theta_m} : \Omega \to \mathbb{R}^{d_m}, \qquad \mathbf{f}_{\theta_f} : \Omega \times \{1, \dots, T\} \to \mathbb{R}^3, \tag{A.1}$$

where θ_m and θ_f denote learnable parameters. In our setting, $\mathbf{m}_{\theta_m}(\mathbf{x})$ encodes the local material properties at position $\mathbf{x}$, such as Young's modulus, Poisson's ratio, and density, while $\mathbf{f}_{\theta_f}(\mathbf{x}, t)$ encodes the external force applied at position $\mathbf{x}$ and time t. This makes the inverse problem explicit: the object of optimization is the pair of field representations (θ_m, θ_f).

Forward Model. The recovered fields are linked to the observed video through a differentiable simulation-and-rendering pipeline. Let $\mathcal{S}$ denote the physics simulator and $\mathcal{R}$ the differentiable renderer. Starting from $\mathcal{G}_0$, the latent state evolves as

$$\mathcal{G}_t = \mathcal{S}\Big(\mathcal{G}_{t-1}, \mathbf{m}_{\theta_m}, \mathbf{f}_{\theta_f}(\cdot, t)\Big), \qquad t = 1, \dots, T, \tag{A.2}$$

and each state is mapped to the image domain by

$$\hat{I}_t = \mathcal{R}(\mathcal{G}_t), \qquad t = 1, \dots, T. \tag{A.3}$$

Together, Eqs. A.2 and A.3 map the latent physical fields to the observed video.

Learning Objective. During training, the forward model produces a rendered video $\hat{\mathcal{V}}(\theta_m, \theta_f) = \{\hat{I}_t\}_{t=1}^{T}$, which should match the observed video $\mathcal{V}$ as closely as possible. The primary supervision is therefore a reconstruction loss $\mathcal{L}_{\text{recon}}$ that compares the rendered video against the observation in image space, encouraging the recovered physical fields to reproduce the visible motion.

In addition, we impose a light regularization term $\mathcal{L}_{\text{reg}}$ on the force field so that the recovered force changes smoothly over space and time rather than exhibiting abrupt, implausible fluctuations. These two terms together give the final objective

$$(\theta_m^*, \theta_f^*) = \arg\min_{\theta_m, \theta_f} \; \mathcal{L}_{\text{recon}} + \lambda_{\text{reg}} \mathcal{L}_{\text{reg}}. \tag{A.4}$$

We do not introduce any generative video prior or score-distillation term: as discussed in Section 3.5, direct video reconstruction together with this light smoothness term is already sufficient to disentangle and recover both fields.

A.2 Details about Our Framework

Gaussian Splatting Representation. Our object representation is based on 3D Gaussian Splatting [11] throughout the pipeline. Given the input video, we first extract the first frame I_1 and apply foreground segmentation so that the reconstruction focuses on the moving object rather than the background. We then use a monocular depth model [20] to predict a depth map for the object region and back-project the foreground pixels into 3D to obtain an initial point cloud. This depth-guided initialization is used together with the first frame to optimize a 3D Gaussian Splatting model

$$\mathcal{G}_0 = \{(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, \alpha_i, \mathbf{c}_i)\}_{i=1}^{N}, \tag{A.5}$$

which serves as the pre-trained object representation in our framework. Since we only reconstruct the foreground object, the static background can be directly composited back during rendering.

Material Field. The material field is represented by a shared causal triplane feature volume [5] defined over the object domain. For each queried spatial location $\mathbf{x}$, we first read out a shared feature vector from the triplane, and then decode different material attributes with independent MLP heads. In particular, Young's modulus and Poisson's ratio are predicted by two separate decoders:

$$\mathbf{z}_m(\mathbf{x}) = \Pi_m(\mathbf{x}), \qquad \Delta E(\mathbf{x}) = h_E(\mathbf{z}_m(\mathbf{x})), \qquad \Delta \nu(\mathbf{x}) = h_\nu(\mathbf{z}_m(\mathbf{x})), \tag{A.6}$$

where Π_m denotes the shared causal triplane representation, and h_E and h_ν are attribute-specific MLP decoders. We initialize the material field with a coarse prior from a Vision-Language Model (VLM), instantiated in our case by Qwen-VL [2], and learn residual corrections on top of it:

$$E(\mathbf{x}) = E_{\text{vlm}} + \Delta E(\mathbf{x}), \qquad \nu(\mathbf{x}) = \nu_{\text{vlm}} + \Delta \nu(\mathbf{x}). \tag{A.7}$$

This lets the VLM provide a physically meaningful starting point while the triplane residual captures spatially varying deviations from that initialization.

Force Field. The force field uses the same triplane-decoder logic, but now the representation is spatio-temporal. We model it as an XYT triplane field [5] that is queried with spatial location and time, and decode the force vector from the queried feature:

$$\mathbf{z}_f(\mathbf{x}, t) = \Pi_f(x, y, t), \qquad \mathbf{f}_{\theta_f}(\mathbf{x}, t) = h_f(\mathbf{z}_f(\mathbf{x}, t)). \tag{A.8}$$

Unlike the material field, the force field does not admit a meaningful VLM initialization. We therefore initialize its output to zero and optimize it from scratch, so the recovered force is entirely driven by video supervision and physical consistency during training.

Physical Simulation Method. To make the Gaussian representation suitable for physics simulation, we take the pre-trained object Gaussians as the starting point for the MPM particle set and augment the interior with additional particles so that the object is no longer represented only on the surface. This gives an initial simulation state

$$\mathcal{P}_0 = \{\mathbf{x}_j^0\}_{j=1}^{M}, \qquad \mathbf{x}_j^0 \in \Omega. \tag{A.9}$$

We then adopt the Material Point Method (MPM) [10] as the physical simulator. Our differentiable MPM implementation is built on NVIDIA Warp [16], which provides GPU-accelerated simulation and gradient computation. For each video frame interval, the simulator performs multiple sub-steps. In each sub-step, it evaluates the current material and force fields, transfers particle quantities to the grid, updates the grid state under internal and external forces,

and transfers the updated motion back to the particles. The full frame-level update is written as

$$\mathcal{P}_t = \mathcal{S}_{\text{MPM}}\Big(\mathcal{P}_{t-1}, \mathbf{m}_{\theta_m}, \mathbf{f}_{\theta_f}(\cdot, t)\Big). \tag{A.10}$$

After the sub-steps for frame t are completed, the updated particles define the displacement and local deformation of the object. We use these particle updates to animate the Gaussians,

$$\boldsymbol{\mu}_i^t = \boldsymbol{\mu}_i^0 + \sum_{j \in \mathcal{N}(i)} w_{ij}\big(\mathbf{x}_j^t - \mathbf{x}_j^0\big), \tag{A.11}$$

where $\mathcal{N}(i)$ denotes particles associated with Gaussian i and w_{ij} are normalized skinning weights. The animated Gaussians are then rendered to produce frame $\hat{I}_t$. Because both the MPM simulation and the Gaussian renderer are differentiable, the whole pipeline can be optimized end-to-end from video supervision.

Optimization and Supervision. At training time, the current fields (θ_m, θ_f) define a simulated trajectory, which is rendered into a video $\hat{\mathcal{V}}$. We optimize a minimal two-term objective consisting of a reconstruction loss $\mathcal{L}_{\text{recon}}$ and a light regularizer $\mathcal{L}_{\text{reg}}$ on the force field:

$$\mathcal{L} = \mathcal{L}_{\text{recon}} + \lambda_{\text{reg}}\,\mathcal{L}_{\text{reg}}. \tag{A.12}$$

The primary supervision is the reconstruction loss, which combines a per-frame L2 loss and an SSIM loss between the rendered and observed frames, both evaluated only inside the foreground mask $\mathcal{M}_t$ of the moving object:

$$\mathcal{L}_{\text{recon}} = \sum_{t=1}^{T}\Big(\big\|\mathcal{M}_t \odot (\hat{I}_t - I_t)\big\|_2^2 + \lambda_{\text{ssim}}\,\big(1 - \text{SSIM}(\mathcal{M}_t \odot \hat{I}_t,\ \mathcal{M}_t \odot I_t)\big)\Big), \tag{A.13}$$

which forces the recovered fields to reproduce the observed motion directly in pixel space. This is the most faithful supervision signal available in the monocular setting, because it measures whether the induced physics explains the actual video rather than an intermediate proxy such as tracked keypoints. We restrict the loss to the foreground mask because only the object deforms while the background is held fixed.

The regularizer $\mathcal{L}_{\text{reg}}$ acts only on the force field and discourages spatially or temporally erratic forces:

$$\mathcal{L}_{\text{reg}} = \sum_{t=1}^{T}\int_{\Omega} \|\nabla \mathbf{f}_{\theta_f}(\mathbf{x}, t)\|_2^2\, d\mathbf{x}, \tag{A.14}$$

which prevents degenerate explanations such as oscillatory force patterns that happen to fit the RGB observations frame-by-frame. We do not regularize the material field separately, since the VLM initialization already provides a sensible prior in space. We also do not introduce any generative video prior: under the training schedule of Section 3.5, this minimal reconstruction-plus-smoothness objective is already sufficient to disentangle and recover both physical fields.

B Data Source and Data Acquisition

Our framework takes a single monocular video as input, which can be acquired from two types of sources:

Figure A.1: Representative data cases. Left: synthetic data with controlled physics parameters (a bouncing basketball and a swaying plant). Right: real-world data captured from everyday scenes (a vibrating spring and a swaying flower). The ⟷ annotations indicate the direction of applied force and the resulting motion.

Synthetic Data. We use physics simulation engines (e.g., Blender [4], Houdini, or custom MPM simulators) to generate synthetic videos with known ground-truth physical parameters. This allows quantitative evaluation of the recovered force and material fields. We plan to generate scenes including deformable objects (e.g., elastic bodies, cloth) under various force conditions (e.g., wind, impact) with diverse material properties.

Real-World Data. We collect real-world videos from publicly available sources depicting common physical phenomena, such as leaves blowing in wind, flowers swaying, and cloth deforming. These videos are captured with a static or slowly moving camera and contain clearly observable object motion driven by external forces. No ground-truth physical annotations are available for these videos; evaluation is performed qualitatively and through downstream tasks such as video generation.

Figure A.1 shows representative examples from both categories. Each case is annotated with ⟷ indicating the direction of the applied external force and the resulting motion. No special hardware or data collection setup is required—a standard RGB camera or publicly available videos suffice.

C Vision-Language Model Initialization Details

We initialize the material field with per-part physical properties produced by a vision-language model (VLM). Given the foreground-segmented first frame of the input video, we prompt the VLM with the template shown in Figure A.2: for each visible foreground part, the model is asked to return a short name, a normalized bounding box, a material class, and three numerical estimates (Young's modulus E, Poisson's ratio ν, density ρ). The output is a JSON array that we parse into per-part property records and propagate to all MPM particles inside each predicted bounding box, with nearest-neighbor fill for any unassigned particles. These per-particle targets are then used to pretrain the material field via MSE regression, as described in Section 3.2.

Figure **??** shows a representative VLM response visualized on an input frame: each predicted bounding box is overlaid on the foreground image and color-coded by the inferred material category, with the inferred (E, ν, ρ) values listed alongside. Particles falling inside each box receive the corresponding material constants as their initialization, providing the spatial pattern that the subsequent MaterialField pretraining and the joint optimization stage refine.

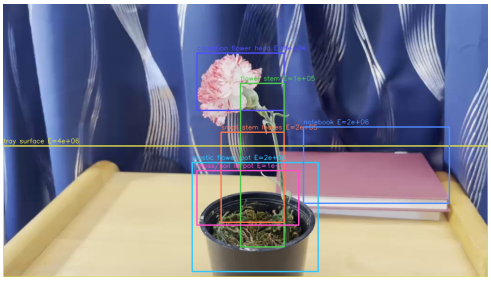
```
You are a physics expert analyzing an image to identify objects and
their material properties.

For each distinct foreground object or object part visible in the
image, provide:
1. A short name (e.g. "ficus leaf", "ceramic pot", "wooden stem")
2. A bounding box in normalized coordinates [x_min, y_min, x_max, y_max]
   where (0,0) is top-left and (1,1) is bottom-right
3. The material class (e.g. "leaf", "ceramic", "wood", "rubber",
   "plastic", "fabric")
4. Estimated Young's modulus E in Pascals (use scientific notation,
   e.g. 3e5)
5. Estimated Poisson ratio nu (between -0.99 and 0.49)
6. Estimated density in kg/m^3 (e.g. leaf ~400, wood ~700,
   ceramic ~2300, metal ~7800)

Return ONLY a JSON array, no other text. Example format:
[
  {"name": "ficus leaf",  "bbox": [0.1, 0.05, 0.9, 0.7],
   "material": "leaf",    "E": 3e5,  "nu": 0.30, "density": 400},
  {"name": "ceramic pot", "bbox": [0.2, 0.6,  0.8, 1.0],
   "material": "ceramic", "E": 7e10, "nu": 0.25, "density": 2300}
]

Important:
- Focus on FOREGROUND objects only, ignore the background.
- Use physically realistic E values (rubber ~1e6 Pa, wood ~1e10 Pa,
  metal ~2e11 Pa, soft tissue ~1e5 Pa).
- Use physically realistic density values (foam ~100, leaf ~400,
  wood ~700, ceramic ~2300, metal ~7800 kg/m^3).
- If you are uncertain about a part, still provide your best estimate.
```

(a) Input image fed to the VLM.



(b) Bounding boxes returned by the VLM.

Figure A.2: Prompt template, input image, and VLM response for material initialization. Left: the prompt template asks the VLM to return, for each foreground part, a name, a normalized bounding box, a material class, and the three physical constants (E, ν, ρ). Concrete order-of-magnitude examples (rubber, wood, ceramic, metal, soft tissue) are embedded in the prompt so that the returned values fall in physically realistic ranges. Right (top): a representative input frame (here, a carnation scene) supplied to the VLM together with the prompt. Right (bottom): the per-part bounding boxes returned by the VLM, drawn on the same frame; particles falling inside each box receive the corresponding (E, ν, ρ) values as their material initialization.

D Training Efficiency

We report the wall-clock time and memory footprint of running our pipeline end-to-end on a single scene, and summarize the aggregate cost over the full benchmark. All numbers are measured on a single NVIDIA GeForce RTX 4090 (24 GB VRAM), as in the main implementation details (Section 4.1). The vision-language model that produces the per-part material initialization (Section 3.2) is consumed as an external API and contributes no local compute or memory cost; we therefore omit it from the breakdown below.

Per-scene VRAM footprint.

- Static 3DGS fitting: ~14 GB.
- Physical field reconstruction (end-to-end): ~16 GB at peak.

The peak occurs during physical field reconstruction and is dominated by the differentiable MPM tape together with the gradients of the material and force fields; both stages fit comfortably within the 24 GB budget of a single 4090.

Per-scene wall-clock time.

- Static 3DGS fitting (~30,000 iterations): ~10 min.
- Physical field reconstruction: ~1.1 s per optimizer iteration, with 150 iterations per transition; the wall-clock cost scales linearly with the number of frames N in the input video.

For the 60-frame setting that produced our main results, the physical field reconstruction completes in ~2.3 hours per scene.

Aggregate cost over the benchmark. Aggregating across the four PhysDreamer [28] scenes, the total optimization cost is approximately 10 GPU-hours on a single RTX 4090. Runs across scenes are independent and can be sharded across multiple GPUs.