PHYSATOBJ: LEARNING PHYSICS-GROUNDED 3D OBJECT INTERACTION FROM MULTI-VIEW VIDEOS

Anonymous authors
Paper under double-blind review

ABSTRACT

Modeling dynamic 3D scenes with interacting objects is crucial for understanding physical environments. However, existing methods struggle to extrapolate future frames because they fail to capture underlying physical dynamics. In this paper, we introduce **PhysAtObj**, a new framework that learns object interaction dynamics directly from multi-view videos without requiring explicit physical properties. The core of PhysAtObj is an object interaction module that identifies dynamic contact regions and directions, and then learns comprehensive impulse-induced and time-agnostic velocity changes during interactions. By relying on the physical states of interacting entities rather than absolute timestamps, the learned dynamics generalize robustly beyond the observed training window. Extensive experiments on PhysInOne and our newly introduced synthetic and real-world datasets demonstrate that PhysAtObj achieves state-of-the-art performance in future frame extrapolation, significantly outperforming baselines.

1 INTRODUCTION

Real-world environments are inherently dynamic, with multiple 3D objects interacting according to physical laws through processes such as sliding, colliding, and rebounding. Developing a model to reason about these dynamics and forecast future outcomes is crucial for autonomous systems to plan their actions effectively. To model dynamic 3D scenes, existing works (Fang et al., 2022; Pumarola et al., 2021; Yang et al., 2024c; Lu et al., 2024) typically extend NeRF (Mildenhall et al., 2020) and 3DGS (Kerbl et al., 2023), focusing primarily on 3D/4D reconstruction and novel view synthesis within an observed time window. While achieving impressive results when interpolating between observed timestamps, they fail to extrapolate future frames, fundamentally because they do not learn the underlying physical priors governing scene's motions. To incorporate physical priors, a line of research (Li et al., 2023b; Cai et al., 2024; Zhong et al., 2024; Xie et al., 2024) seeks to infer physical properties or dynamics directly from visual observations and use simulators to infer future frames. Despite yielding excellent results, they are often restricted to specific types of single objects.

Recently, a new line of work represented by FreeGave (Li et al., 2025b) and TRACE (Li et al., 2025a) learns 3D velocity fields directly from multi-view videos, producing smooth Gaussian particle trajectories that satisfy the essential divergence-free physical constraint. While achieving state-of-the-art performance in future frame extrapolation on multiple datasets, both can only extrapolate limited motion patterns. They remain unable to predict future object interactions, primarily because they fail to learn the fundamental causal dynamics, *e.g.*, the collision-induced velocity changes governing a bouncing ball. More recently, several works (Wang et al., 2026c; Li et al., 2026a; Gao et al., 2025) attempt to learn force fields from videos. However, their proposed force fields are dependent on timestamps, making them unable to extrapolate the learned forces for future object interactions. Furthermore, these methods often need object properties such as mass to be provided *a priori*.

In this paper, we aim to introduce a new framework for modeling dynamic 3D scenes which comprise multiple interacting objects during the observed time window, just from multi-view videos without requiring object properties such as materials or masses. Ultimately, our approach enables the prediction of future frames where these dynamic interactions continue to unfold. Achieving this is highly challenging, as it requires not only modeling dynamic objects' geometry, appearance, and motion patterns, but also learning the underlying physical factors that govern object interactions.

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.

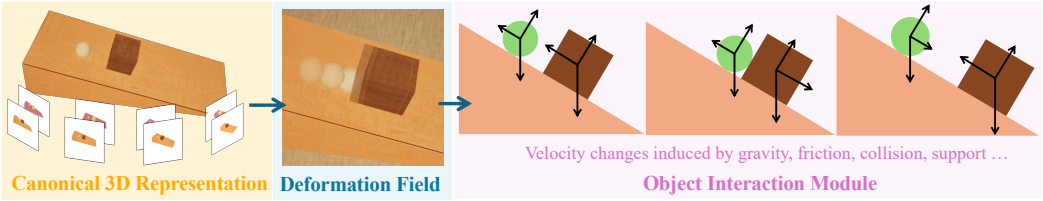

Figure 1: The general workflow and components of our framework.

As illustrated in Figure 1, our framework comprises three primary components: 1) **a canonical 3D representation module**, which leverages 3DGS (Kerbl et al., 2023) to learn a set of 3D Gaussians augmented with object codes at a canonical timestamp; 2) **an auxiliary deformation field** adopted to capture per-object motion information within the observed time window; and 3) **an object interaction module** that systematically models the physical interactions among multiple objects.

The core of our framework is the object interaction module. Particularly, at any given timestamp, it first identifies the potential contact region and direction between any two objects, and then learns comprehensive impulse-induced velocity changes during interaction. A key insight of our approach is that the learning of these velocity changes should be entirely time-agnostic, relying only on the **phys**ical states of inter**act**ing **obj**ects. This allows the learned interaction dynamics to generalize well beyond the observed time window, effectively predicting correct object interactions during future extrapolation. Our method is named **PhysAtObj** and our contributions are:

- We propose a new framework that models dynamic 3D scenes of interacting objects directly from multi-view videos. Without requiring *a priori* knowledge of object masses or materials, our method explicitly captures the object geometry, appearance, and physical dynamic states.
- We introduce an object interaction module that identifies contact regions and systematically learns time-agnostic impulse-induced velocity changes during interaction, enabling the learned dynamics to generalize beyond the observed time window for accurate future extrapolation.
- Extensive experiments demonstrate that PhysAtObj achieves state-of-the-art results on multiple datasets, surpassing existing baselines that merely extend observed motion patterns.

2 Related Work

Dynamic 3D Modeling: Existing approaches for dynamic 3D scene reconstruction and novel view synthesis generally extend static representations, (*e.g.*, SDF (Park et al., 2019), NeRF (Mildenhall et al., 2020), and 3DGS (Kerbl et al., 2023)) by introducing a temporal dimension (Pumarola et al., 2021; Park et al., 2021; Tretschk et al., 2021; Gao et al., 2021; Du et al., 2021; Tian et al., 2023; Liu et al., 2023; Fridovich-Keil et al., 2023; Cao & Johnson, 2023; Liu et al., 2022; Fang et al., 2022; Yao et al., 2024; Yang et al., 2024a; Im et al., 2024; Zhan et al., 2024). Although these methods successfully model the motion of rigid and deformable targets and achieve exceptional novel view synthesis results, especially with 3DGS (Lin et al., 2024; Lu et al., 2024; Qian et al., 2024; Yang et al., 2024c; Lu et al., 2024; Bae et al., 2024; Li et al., 2024b; Yang et al., 2024b; Wu et al., 2024; Mihajlovic et al., 2024; Lei et al., 2025; Liu et al., 2025; Park et al., 2025; Xu et al., 2025; Chen et al., 2025b; Lin et al., 2026; Yeom et al., 2026), their capabilities are limited to interpolation within the observed time window. Because they are optimized solely to fit visual observations rather than learning underlying physical priors, they remain unable to predict physically plausible future frames. In contrast, our PhysAtObj is designed to learn comprehensive impulse-induced velocity changes during object interactions, ensuring that the extrapolated future frames adhere to physical dynamics.

Dynamic Video Generation: Driven by advancements in diffusion and flow matching techniques (Ho et al., 2020; Lipman et al., 2023; Esser et al., 2024), recent video generation models (Höppe et al., 2022; Ho et al., 2022; Voleti et al., 2022; Ni et al., 2023; Yu et al., 2023; Chen et al., 2024; Blattmann et al., 2023; Yang et al., 2025; Wan, 2025) excel at synthesizing photorealistic frames. Despite this high visual fidelity, they frequently produce physically implausible motions, such as objects falling upward or suddenly shifting velocity, as also discussed in Motamed et al. (2025); Kang et al. (2025). This shortcoming stems primarily from a reliance on standard pixel-level reconstruction losses, which are insufficient to ensure that the learned latent representations capture underlying physical dynamics.

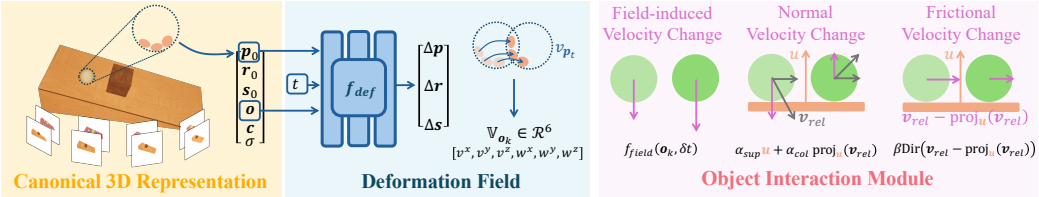

Figure 2: The leftmost block depicts the canonical 3D representation module, the middle block outlines the auxiliary deformation field, and the rightmost block presents our object interaction module designed to learn comprehensive impulse-induced velocity changes.

3D Physics Learning and Simulation: Early approaches to learning physical priors (Liang et al., 2019; Jonathan et al., 2020; Allen et al., 2023; Driess et al., 2022; Xue et al., 2023; Shi et al., 2024; Whitney et al., 2024; Rubanova et al., 2024) use graph networks to simulate particle-level dynamics within objects but rely on handcrafted graph constructions, limiting their scalability. The majority of recent methods seek to estimate physical properties, aka system identification, from visual observations and subsequently employ simulation systems such as MPM (Hu et al., 2018; Li et al., 2023b; 2024a; Kaneko, 2024; Xie et al., 2024; Qiu et al., 2024; Cao et al., 2024; Cai et al., 2024; Li et al., 2024a; Fu et al., 2024; Borycki et al., 2024; Tan et al., 2024; Le et al., 2025; Chen et al., 2025a; Vasile et al., 2025; Lin et al., 2025; Wang et al., 2026d; Lv et al., 2026; Foo et al., 2026; Huang et al., 2026a;b; Lei et al., 2026; Liu et al., 2026; Feng et al., 2026), Spring-Mass (Zhong et al., 2024; Jiang et al., 2025; Wang et al., 2026a), and others (Feng et al., 2024b;a; Jia et al., 2025; Wang et al., 2026b) to animate objects or predict object future dynamics. To further enhance physical plausibility, visual priors from video generation models have also been incorporated in recent frameworks (Liu et al., 2024a;b; Zhang et al., 2024; Huang et al., 2025). Despite their impressive results, these simulation-based methods are generally limited to specific object materials and often rely on privileged information, such as depth measurements and physical priors. Moreover, they primarily focus on modeling single-object dynamics, often assuming access to known object boundaries. Another line of research (Raissi et al., 2020; Chu et al., 2022; Baieri et al., 2023; Hao et al., 2023; Cho et al., 2024; Li et al., 2023a; Zhang et al., 2025; Wang et al., 2026c; Quan et al., 2026; Li et al., 2026b;a; Zhobro et al., 2025) integrates physics priors into learning frameworks by adding PDEs/ODEs into loss functions or neural networks. However, they often struggle to model object interactions that cannot be readily described by differential equations.

3 PHYSATOBJ

Given multi-view RGB videos with known camera poses and intrinsics, denoted as $\{\{\mathcal{I}_0^1 \cdot \cdot \mathcal{I}_0^C\} \cdot \cdot \{\mathcal{I}_T^1 \cdot \cdot \mathcal{I}_T^C\}\}$, where C represents the total number of cameras and T the greatest timestamp in training. These videos capture multiple objects interacting with each other. We assume the availability of object masks for each image at time $t = 0$, denoted as $\{\mathcal{O}_0^1 \cdot \cdot \mathcal{O}_0^C\}$, which can be easily obtained by existing models like SAM. Our method aims to learn the scene's 3D geometry, appearance and objectness through a canonical 3D representation module elaborated in Section 3.1, the motion information of training time through an auxiliary deformation field discussed in Section 3.2, and, most importantly, the interaction dynamics through an object interaction module detailed in Section 3.3.

3.1 CANONICAL 3D REPRESENTATION MODULE

As shown in the leftmost block of Figure 2, this module builds upon 3DGS to learn a set of 3D Gaussian kernels $\boldsymbol{P}_0$, which represent the canonical scene geometry and appearance at time $t = 0$. Each kernel is parameterized by a 3D position $\boldsymbol{p}_0$, scaling $\boldsymbol{s}_0$, covariance matrix converted from quaternion $\boldsymbol{r}_0$, opacity σ, and color $\boldsymbol{c}$ converted from spherical harmonics. Following dynamic 3D Gaussians (Yang et al., 2024c), we assume the opacity σ and color $\boldsymbol{c}$ of each kernel remain constant over time. Furthermore, we follow GassianGrouping (Ye et al., 2024) to associate a learnable object code $\boldsymbol{o}$ to each kernel, which can be converted into a one-hot vector to indicate object identity. Given the C multi-view images at time $t = 0$, we initialize canonical kernels and project them into camera space to render 2D images and object masks. All Gaussian properties are jointly optimized using standard image losses (ℓ_1 and ℓ_{ssim}) and cross-entropy based segmentation loss ℓ_{seg} as follows.

$$\underbrace{\{\cdots(\boldsymbol{p}_0, \boldsymbol{r}_0, \boldsymbol{s}_0, \boldsymbol{o}, \sigma, \boldsymbol{c})\cdots\}}_{\boldsymbol{P}_0} \underset{\ell_1+\ell_{\text{ssim}}+\ell_{\text{seg}}}{\overset{\text{render}}{\rightleftarrows}} \{\mathcal{I}_0^1/\mathcal{O}_0^1 \cdots \mathcal{I}_0^c/\mathcal{O}_0^c \cdots \mathcal{I}_0^C/\mathcal{O}_0^C\} \tag{1}$$

3.2 Auxiliary Deformation Field

As shown in the middle block of Figure 2, this module aims to learn the motion information across the training time T. For simplicity, we follow DefGS to learn a continuous deformation field. Particularly, we feed each canonical Gaussian $\boldsymbol{p}_0$, its object code $\boldsymbol{o}$, timestamp t, into an MLP-based network f_{def}, predicting the corresponding position displacement $\Delta\boldsymbol{p}$, and the change of quaternion $\Delta\boldsymbol{r}$ and scaling $\Delta\boldsymbol{s}$ from timestamp 0 to t. The deformed Gaussians $\boldsymbol{P}_t$ at time t are then computed by transporting canonical Gaussians as follows, where the operators $\odot$ and $\circ$ follow DefGS.

$$(\Delta\boldsymbol{p}, \Delta\boldsymbol{s}, \Delta\boldsymbol{r}) = f_{def}(\boldsymbol{p}_0, \boldsymbol{o}, t), \boldsymbol{P}_t \leftarrow \{\cdot\cdot(\boldsymbol{p}_t = \boldsymbol{p}_0 + \Delta\boldsymbol{p}, \boldsymbol{s}_t = \boldsymbol{s}_0 \odot \Delta\boldsymbol{s}, \boldsymbol{r}_t = \boldsymbol{r}_0 \circ \Delta\boldsymbol{r}, \boldsymbol{o}, \sigma, \boldsymbol{c})\cdots\} \tag{2}$$

This deformation field is simply optimized by rendering all Gaussians $\boldsymbol{P}_t$ into image space and comparing against the observed images at time t in training set. Although the optimized deformation field cannot predict future motions, as extrapolation falls outside the training domain of time t, it captures movement of all Gaussians across training period T. If consolidated at the object level, this motion information becomes highly valuable for learning object interaction dynamics.

To this end, we group all Gaussians $\boldsymbol{P}_t$ into K objects based on their learned object codes. For a specific k^{th} object $\boldsymbol{o}_k$, we approximate the velocity $\boldsymbol{v}_{\boldsymbol{p}_t}$ for each of its Gaussians as follows:

$$\boldsymbol{v}_{\boldsymbol{p}_t} = \frac{\Delta\boldsymbol{p} - \Delta\boldsymbol{p}'}{\Delta t}, \Delta\boldsymbol{p} \leftarrow f_{def}(\boldsymbol{p}_0, \boldsymbol{o}_k, t) \text{ and } \Delta\boldsymbol{p}' \leftarrow f_{def}(\boldsymbol{p}_0, \boldsymbol{o}_k, t - \Delta t),\ \Delta t \text{ is a small interval.} \tag{3}$$

Next, we compute six basic velocity components, denoted as $\mathbb{V}_{\boldsymbol{o}_k} \leftarrow [v^x, v^y, v^z, w^x, w^y, w^z]$, shared by the whole object $\boldsymbol{o}_k$ from all its Gaussians' velocities using least squares optimization as follows.

$$\mathbb{V}_{\boldsymbol{o}_k} = \underset{\mathbb{V}_{\boldsymbol{o}_k}}{\arg\min} ||\mathbb{V}_{\boldsymbol{o}_k} \cdot \mathcal{B}(\boldsymbol{p}_t) - \boldsymbol{v}_{\boldsymbol{p}_t}||, \quad \mathcal{B}(\boldsymbol{p}_t) = \begin{bmatrix} 1 & 0 & 0 & 0 & p_t^z & -p_t^y \\ 0 & 1 & 0 & -p_t^z & 0 & p_t^x \\ 0 & 0 & 1 & p_t^y & -p_t^x & 0 \end{bmatrix}^T \tag{4}$$

where $\boldsymbol{p}_t$ here represents each single Gaussian within $\boldsymbol{o}_k$, and $p_t^x/p_t^y/p_t^z$ are coordinates of $\boldsymbol{p}_t$. This step consolidates per-Gaussian motion into the overall motion of the object. We also retain the remaining local velocity as $\boldsymbol{v}_{\boldsymbol{p}_t}^{local} = \boldsymbol{v}_{\boldsymbol{p}_t} - \mathbb{V}_{\boldsymbol{o}_k} \cdot \mathcal{B}(\boldsymbol{p}_t)$ to describe its deformation. The shared basic components with $\mathcal{B}$ provide a general object-level divergence-free velocity field, following FreeGave, while the local velocity also allows potential local deformation.

3.3 Object Interaction Module

At any timestamp t, the motion of an object $\boldsymbol{o}_k$ consists of its shared basic velocity components $\mathbb{V}_{\boldsymbol{o}_k}$ and per-Gaussian local velocities $\boldsymbol{v}_{\boldsymbol{p}_t}^{local}$. External fields and contact interactions change its overall motion, while local deformation can persist even after contact ends. These changes depend on the current physical state rather than the absolute time t. With this insight, our object interaction module first identifies contact regions, predicts local contact responses, and then propagates them with the current deformation state to update the object Gaussians' motion, as illustrated in Figure 2.

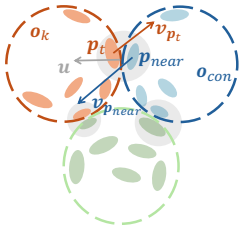

Figure 3: Object contact regions and directions.

Object Contact Region and Direction: As shown in Figure 3, regarding an object $\boldsymbol{o}_k$, for each of its Gaussians $\boldsymbol{p}_t$, we search for the nearest Gaussian $\boldsymbol{p}_{near}$ that does not belong to $\boldsymbol{o}_k$ within the entire dynamic 3D scene. If the Euclidean distance between the two Gaussian centers is less than a threshold δd, then $\boldsymbol{p}_t$ is identified as a contact Gaussian. Otherwise, $\boldsymbol{p}_t$ is a non-contact Gaussian. Once all contact Gaussians for $\boldsymbol{o}_k$ are identified, we group them into distinct contact regions, where each region $\mathcal{C}$ corresponds to an interaction with one neighboring object. If $\boldsymbol{o}_k$ interacts with multiple objects simultaneously, it will yield multiple contact regions. For each contact region $\mathcal{C}$ on $\boldsymbol{o}_k$, we obtain its contact direction $\boldsymbol{u}$ by computing a normal vector of the corresponding contact Gaussians oriented toward $\boldsymbol{o}_k$. More details are in Appendix A.1.

After identifying contact regions and directions, we naturally have the corresponding contacted regions on neighboring objects. Furthermore, the motion states of all interacting objects have already been established in Section 3.2. Utilizing this critical information, we aim to learn comprehensive impulse-induced velocity changes during an interaction.

We first model three sources of velocity changes: 1) a field-induced velocity change driven by environmental forces like gravity; 2) a normal velocity change driven by normal contact impulses along the contact normal; and 3) a frictional velocity change driven by tangential friction impulses acting perpendicularly to the contact normal. We then combine these responses with the current object states through a shared propagation network to model rigid and deformable motions as follows.

1) Field-induced Velocity Change: Regarding an object $\boldsymbol{o}_k$ at any timestamp t, we feed its object code together with a predefined and fixed small time interval δt into an MLP-based network f_{field}, predicting the change of six basic velocity components $\delta\mathbb{V}_f$ of the entire object $\boldsymbol{o}_k$ as follows:

$$\delta\mathbb{V}_f = f_{field}(\boldsymbol{o}_k, \delta t) \tag{5}$$

Essentially, f_{field} aims to learn the global velocity change induced by external fields over δt.

2) Normal Velocity Change: For a target object $\boldsymbol{o}_k$ and its contacting object $\boldsymbol{o}_{con}$, normal contact responses arise from collision and/or support along the contact direction $\boldsymbol{u}$. At each contact Gaussian $\boldsymbol{p}_c \in \mathcal{C}$, we compute the relative velocity $\boldsymbol{v}_{rel} = \boldsymbol{v}_{\boldsymbol{p}_c} - \boldsymbol{v}_{\boldsymbol{p}_{near}}$. The collision response is activated for approaching contacts, where $\boldsymbol{v}_{rel} \cdot \boldsymbol{u} \leq 0$, while support acts at established contacts. We denote the support response magnitude by α_{sup} and the nonnegative collision response coefficient by α_{col}. An MLP network f_{normal} predicts these coefficients:

$$\alpha_{sup}, \alpha_{col} = f_{normal}(\boldsymbol{o}_k, \boldsymbol{o}_{con}, \delta t) \tag{6}$$

For a contact direction $\boldsymbol{u}$, let $\mathrm{proj}_{\boldsymbol{u}}(\boldsymbol{v}_{rel}) = (\boldsymbol{v}_{rel} \cdot \boldsymbol{u})\boldsymbol{u}$ denotes the normal component of the relative velocity. The resulting normal velocity response at $\boldsymbol{p}_c$ is $\boldsymbol{j}^n_{\boldsymbol{p}_c} = \alpha_{sup}\boldsymbol{u} - \alpha_{col}\,\mathrm{proj}_{\boldsymbol{u}}(\boldsymbol{v}_{rel})$, with the collision term applied only to approaching contacts and $\alpha_{col} \geq 0$. We retain this response at each contact Gaussian, together with its position and normal direction, for the propagation step below.

3) Frictional Velocity Change: Similarly, frictional contact responses oppose the relative sliding direction $\mathrm{Dir}\big(\boldsymbol{v}_{rel} - \mathrm{proj}_{\boldsymbol{u}}(\boldsymbol{v}_{rel})\big)$. We feed the object codes into another MLP network $f_{friction}$ to predict the nonnegative friction response magnitude β:

$$\beta = f_{friction}(\boldsymbol{o}_k, \boldsymbol{o}_{con}, \delta t) \tag{7}$$

The resulting tangential velocity response at a contact Gaussian is $\boldsymbol{j}^r_{\boldsymbol{p}_c} = -\beta\,\mathrm{Dir}\big(\boldsymbol{v}_{rel} - \mathrm{proj}_{\boldsymbol{u}}(\boldsymbol{v}_{rel})\big)$, where $\mathrm{Dir}(\cdot)$ denotes the unit direction and returns zero for a zero vector. As with the normal response, we preserve its spatial distribution rather than first reducing it to a single vector for the contact region. Here, $\boldsymbol{j}^n_{\boldsymbol{p}_c}$ and $\boldsymbol{j}^r_{\boldsymbol{p}_c}$ denote velocity responses over δt, without requiring explicit mass information.

4) State-aware Velocity Propagation: Given the normal and frictional velocity changes at contact Gaussians, we aim to learn how these interactions affect an object's overall motion and deformation. For each contact Gaussian of an object $\boldsymbol{o}_k$, we denote its contact responses, spatial location, and normal direction as $\mathcal{J}_{\boldsymbol{o}_k} \leftarrow (\boldsymbol{p}_c, \boldsymbol{j}^n_{\boldsymbol{p}_c}, \boldsymbol{j}^r_{\boldsymbol{p}_c}, \boldsymbol{u})$. For each non-contact Gaussian, let $\boldsymbol{p}_0$ and $\boldsymbol{p}_t$ denote its canonical and current positions, respectively. The shared rigid velocity component at time t is $\mathbb{V}_{\boldsymbol{o}_k}(t)$, and $\boldsymbol{v}^{local}_{\boldsymbol{p}_t}$ is its velocity relative to this rigid motion, as defined in Section 3.2. We collect these quantities into the object state $\mathcal{S}_{\boldsymbol{o}_k}(t)$:

$$\mathcal{S}_{\boldsymbol{o}_k}(t) = \Big(\mathbb{V}_{\boldsymbol{o}_k}(t), \big\{\big(\boldsymbol{p}_0, \boldsymbol{p}_t, \boldsymbol{v}^{local}_{\boldsymbol{p}_t}\big)\big\}_{\boldsymbol{p}_t \in \boldsymbol{o}_k}\Big) \tag{8}$$

We denote the predicted contact-induced change of the shared rigid velocity component by $\delta\mathbb{V}^{con}_{\boldsymbol{o}_k}$, and the local velocity change of each Gaussian by $\delta\boldsymbol{v}^{local}_{\boldsymbol{p}_t}$. Conditioned on the object code, a propagation network f_{prop} jointly predicts these changes from the contact responses, object state, field-induced velocity change, and time interval:

$$\Big(\delta\mathbb{V}^{con}_{\boldsymbol{o}_k}, \big\{\delta\boldsymbol{v}^{local}_{\boldsymbol{p}_t}\big\}_{\boldsymbol{p}_t \in \boldsymbol{o}_k}\Big) = f_{prop}\left(\mathcal{J}_{\boldsymbol{o}_k}, \mathcal{S}_{\boldsymbol{o}_k}(t), \boldsymbol{o}_k, \delta\mathbb{V}_f, \delta t\right) \tag{9}$$

The local velocity changes are designed to contain deformation. Denoting the total velocity change of a Gaussian by $\delta\boldsymbol{v}_{\boldsymbol{p}_t}$, we combine the field, rigid contact, and local responses as follows:

$$\delta\boldsymbol{v}_{\boldsymbol{p}_t} = \big(\delta\mathbb{V}_f + \delta\mathbb{V}^{con}_{\boldsymbol{o}_k}\big) \cdot \mathcal{B}(\boldsymbol{p}_t) + \delta\boldsymbol{v}^{local}_{\boldsymbol{p}_t} \tag{10}$$

In the absence of contact, the contact-induced rigid velocity change is zero, while the local branch continues to evolve with the object state. We apply this propagation at every integration step, referring to the canonical geometry. More details are provided in Appendix A.2&A.3.

3.4 TRAINING

With the trained canonical 3D representation in Section 3.1 and the deformation field in Section 3.2, we now discuss how to train the object interaction module in Section 3.3.

- **First**, with a small δt, we select two timestamps t' and t, where t is sampled from the training split and t' is set as $t - \delta t$. Using the trained deformation field f_{def}, we obtain all Gaussians $\boldsymbol{P}_{t'}$

according to Equation 2. For each object $\boldsymbol{o}_k$, we compute the per-Gaussian velocities and their shared rigid components using Equations 3 and 4, retaining the residual local velocities. Together with the canonical geometry, these quantities form the object state $\mathcal{S}_{\boldsymbol{o}_k}(t')$.

- **Second**, for an object $\boldsymbol{o}_k$, we identify its contacting objects. The networks f_{field}, f_{normal}, and $f_{friction}$ provide the field-induced velocity change $\delta\mathbb{V}_f$ and the per-contact-Gaussian responses using Equations 5–8. We collect the contact responses, positions, and normal directions into $\mathcal{J}_{\boldsymbol{o}_k}$, and pass them together with $\mathcal{S}_{\boldsymbol{o}_k}(t')$, the object code, $\delta\mathbb{V}_f$, and δt to f_{prop} according to Equation 9.
- **Lastly**, we combine the predicted velocity changes using Equation 10. Starting from the current velocities, we transport Gaussians $\boldsymbol{P}_{t'}$ to time t using the motion update described in Appendix A.4. We then render the updated Gaussians and compare them against the training images at time t using standard 3DGS losses. For multi-step updates, we recompute the contact responses and apply f_{prop} to the evolving object state at every integration step.

4 EXPERIMENTS

Datasets: While datasets for dynamic 3D reconstruction or novel view interpolation support rendering, they are unsuited for physics-grounded future extrapolation. Datasets such as (Liu et al., 2024b; Xie et al., 2024; Zhong et al., 2024; Vasile et al., 2025; Huang et al., 2026b; Tan et al., 2024; Lei et al., 2026) focus on parameter estimation for downstream simulation and animation rather than direct interaction forecasting. Benchmarks like (Li et al., 2023b; Cao et al., 2024; Li et al., 2025a;b) are tailored to single-object dynamics, making them inadequate for evaluating multi-object interactions. The most relevant existing dataset is **PhysInOne** (Zhou et al., 2026), the largest and most diverse dataset for visual physics learning. To comprehensively evaluate our method, we further collect two new synthetic datasets and one real-world dataset: 1) **Moving-Static Object Interaction Dataset**, 2) **Moving-Moving Object Interaction Dataset**, and 3) **Real-world Object Interaction Dataset**.

1) Moving-Static Object Interaction Dataset: This dataset comprises 4 distinct dynamic 3D scenes, each featuring a moving object interacting with a static object or the background, which have unique but unknown physical properties such. For each 3D scene, we capture RGB images from 20 different viewing angles over an average duration of 3.75 seconds at 30 fps, resulting in 112 frames per viewpoint. We also collect ground truth object masks for the first timestamp. To evaluate future frame extrapolation, we formulate the training split using the first 50% of the time window from 17 randomly selected viewpoints (*i.e.*, 17*56 frames). The test split consists of the remaining 50% of the time window across all 20 viewpoints (*i.e.*, 20*56 frames). More details are in Appendix A.5.

2) Moving-Moving Object Interaction Dataset: This dataset comprises 6 distinct dynamic 3D scenes, each of which displays a moving object interacting with other moving objects. Similarly, for each 3D scene, we capture RGB images from 20 viewpoints over an average duration of 3.33 seconds at 30fps, resulting in 100 frames per viewpoint. The first 50% of the time window from 17 viewpoints is used for training, while the remaining 50% of the time window across all 20 viewpoints is left for testing future frame extrapolation. More details are provided in Appendix A.6.

3) Real-world Object Interaction Dataset: We capture 7 dynamic 3D scenes: 4 featuring a moving-static object interaction, and 3 featuring a moving-moving object interaction. For each scene, we record 120 frames from 20 different viewpoints. The training split consists of the first 50% of the frames from 17 cameras, while the remaining 50% of the frames across all 20 cameras are reserved for evaluating future frame extrapolation. For training views at the first timestamp, we use SAM (Ravi et al., 2024) to segment objects as our training object labels. More details are in Appendix A.7.

4) PhysInOne (Zhou et al., 2026): Specifically, we select 7 representative dynamic 3D scenes from the training partition characterized by rich rigid and deformable object collisions and dynamic contact events. Following the standard PhysInOne protocol, each scene is captured by 12 static camera views, allocating 9 views for training and 3 held-out views for validation. More details are in Appendix A.8.

Baselines: We compare our approach against the following baselines: **1) DefGS** (Yang et al., 2024c), a strong deformable 3DGS method for modeling dynamic 3D scenes and novel view interpolation, which inherently lacks future extrapolation capabilities. **2) DefGS**$_{nvfi}$, a hybrid baseline combining DefGS with the PINN-based NVFi (Li et al., 2023a), leveraging DefGS for novel view synthesis and NVFi for future extrapolation. **3) FreeGave** (Li et al., 2025b), a state-of-the-art method for future frame prediction, but it is limited to extrapolating motion patterns observed during training. **4)**

Table 1: Quantitative results of all methods for future frame extrapolation on all four datasets.

Moving-Static Object Interaction Dataset

	AP↑	PQ↑	F1↑	Pre↑	Rec↑	mIoU↑	mPSNR↑	mSSIM↑	mLPIPS↓	PSNR↑	SSIM↑	LPIPS↓
DefGS (Yang et al., 2024c)	27.193	21.953	35.440	34.473	37.135	28.343	16.001	0.667	0.637	27.643	0.882	0.179
DefGS$_{nvfi}$	26.317	22.917	34.150	31.588	45.203	27.003	17.927	0.686	0.669	27.728	0.878	0.191
ODE-GS (Wang et al., 2026c)	30.238	26.983	33.845	33.465	34.818	30.280	17.018	0.702	0.613	27.186	0.808	0.250
TRACE (Li et al., 2025a)	30.818	25.280	38.038	38.273	38.150	31.640	16.699	0.672	0.654	28.791	**0.905**	0.173
FreeGave (Li et al., 2025b)	39.723	34.848	45.440	46.623	45.603	40.105	18.217	0.697	0.600	30.015	0.884	0.169
PhysAtObj (Ours)	**66.760**	**59.615**	**77.278**	**77.648**	**77.140**	**67.255**	**21.344**	**0.757**	**0.385**	**31.475**	0.884	**0.163**
PhysAtObj$_{multiScenes}$ (Ours)	*57.458*	*49.322*	*69.840*	*70.663*	*70.093*	*59.898*	*18.782*	*0.720*	*0.501*	*30.172*	*0.883*	*0.167*

Moving-Moving Object Interaction Dataset

	AP↑	PQ↑	F1↑	Pre↑	Rec↑	mIoU↑	mPSNR↑	mSSIM↑	mLPIPS↓	PSNR↑	SSIM↑	LPIPS↓
DefGS (Yang et al., 2024c)	48.823	41.600	60.573	60.317	61.427	50.802	16.834	0.663	0.514	27.502	0.872	0.210
DefGS$_{nvfi}$	71.017	62.358	81.183	82.297	81.510	72.035	17.875	0.690	0.505	29.024	**0.875**	0.208
ODE-GS (Wang et al., 2026c)	51.323	39.445	65.553	70.317	62.428	53.371	15.992	0.654	0.563	26.647	0.804	0.269
TRACE (Li et al., 2025a)	71.527	60.548	81.780	85.820	78.863	72.201	19.058	0.711	0.386	29.721	0.859	0.193
FreeGave (Li et al., 2025b)	65.893	55.135	78.015	81.653	75.473	67.288	18.072	0.680	0.470	28.683	0.864	0.204
PhysAtObj (Ours)	**78.574**	**70.740**	**87.229**	**88.658**	**82.188**	**79.403**	**21.161**	**0.733**	**0.351**	**30.691**	0.869	**0.190**

Real-world Object Interaction Dataset

	AP↑	PQ↑	F1↑	Pre↑	Rec↑	mIoU↑	mPSNR↑	mSSIM↑	mLPIPS↓	PSNR↑	SSIM↑	LPIPS↓
DefGS (Yang et al., 2024c)	32.603	25.053	45.057	47.910	46.417	34.728	14.280	0.629	0.658	24.981	0.924	0.126
DefGS$_{nvfi}$	33.817	22.932	47.065	45.613	34.553	28.640	14.943	0.645	0.654	25.387	**0.926**	0.124
ODE-GS (Wang et al., 2026c)	47.321	39.723	61.386	61.662	62.750	50.080	14.712	0.641	0.698	23.639	0.898	0.170
TRACE (Li et al., 2025a)	47.620	39.042	60.667	61.778	59.961	50.463	15.052	0.645	0.652	25.086	0.920	0.134
FreeGave (Li et al., 2025b)	48.318	39.003	62.943	66.108	61.257	51.251	15.886	**0.682**	**0.568**	25.604	0.924	**0.120**
PhysAtObj (Ours)	**59.977**	**51.998**	**72.805**	**72.355**	**73.955**	**61.045**	**16.394**	0.651	0.577	**25.992**	0.917	0.122

PhysInOne Dataset

	AP↑	PQ↑	F1↑	Pre↑	Rec↑	mIoU↑	mPSNR↑	mSSIM↑	mLPIPS↓	PSNR↑	SSIM↑	LPIPS↓
DefGS (Yang et al., 2024c)	55.573	49.791	60.767	60.041	63.566	54.773	17.763	0.748	0.346	21.940	0.849	0.193
DefGS$_{nvfi}$	71.739	66.484	79.623	78.873	81.051	72.751	20.149	0.778	0.323	**24.998**	**0.874**	**0.158**
ODE-GS (Wang et al., 2026c)	69.773	64.897	77.243	74.274	82.906	70.836	18.976	0.692	0.422	22.377	0.723	0.330
TRACE (Li et al., 2025a)	79.193	74.239	83.770	84.257	85.006	79.136	19.960	**0.782**	0.309	24.814	0.870	0.161
FreeGave (Li et al., 2025b)	75.583	71.799	79.909	79.709	80.306	75.780	20.146	0.767	0.327	23.932	0.860	0.174
PhysAtObj (Ours)	**81.383**	**77.225**	**87.432**	**87.356**	**87.706**	**82.271**	**20.556**	**0.782**	**0.286**	24.857	0.872	0.161

TRACE (Li et al., 2025a), which learns a translation-rotation dynamics system for future prediction but fails to model object interactions. **5) ODE-GS** (Wang et al., 2026c), a very recent approach utilizing neural ODEs for dynamic modeling, which lacks the capacity to handle interacting objects.

Metrics: For future frame extrapolation, to assess object mask quality, we employ standard segmentation metrics: Average Precision (**AP**), Panoptic Quality (**PQ**), **F1**-score, Precision (**Pre**), Recall (**Rec**), and mean Intersection over Union (**mIoU**). To evaluate the visual quality of the extrapolated frames, we report **PSNR**, **SSIM**, and **LPIPS** across the full images. Additionally, we compute these metrics exclusively within the foreground dynamic object masks, denoted as **mPSNR**, **mSSIM**, and **mLPIPS**. Notably, these masked metrics are of greater significance, as they specifically reflect the synthesis quality of the interacting objects. We evaluate the computation cost in Appendix A.9.

4.1 Main Results of Future Frame Extrapolation

For fair comparisons, all methods are trained in a scene-specific manner across all datasets with the same number of training images and object masks. For evaluation, future frames are extrapolated by sequentially stepping forward from the last observed timestamp, using the trained object interaction module on top of the deformation field. Table 1 and Figure 4 compare all methods on the four datasets for future frame extrapolation. It can be seen that:

- Regarding object segmentation on the extrapolated future frames, our method significantly outperforms all baselines by large margins across most datasets. This demonstrates that without explicitly modeling the interaction dynamics between entities like us, existing baselines struggle to accurately predict and capture object-level trajectories beyond the observed training window.
- Furthermore, regarding the visual quality of dynamic objects, our method consistently surpasses all baselines, achieving the best mPSNR, mSSIM, and mLPIPS scores in most datasets. This highlights that our method can successfully track and render moving objects with high visual fidelity, preserving their structural integrity and fine details during interactions.
- Regarding full-image rendering metrics (PSNR, SSIM, and LPIPS), we notice that the performance gap between our method and baselines is narrower. This is because full-image scores are

heavily dominated by static backgrounds, which most baseline methods are already capable of reconstructing with relatively high visual fidelity.

4.2 Generalization and Analysis of Object Interaction Module

To evaluate whether our object interaction module can truly learn physical priors, we further conduct the following generalization experiments.

Generalization to Unseen Geometries and Different Initial Velocities. To test whether the module overfits to specific spatial scales or initial dynamics, we evaluate zero-shot inference on unseen geometries and initial velocities. As shown in Figure 5, we change the initial velocity of the ball from downwards to upwards, and replace the *flat ground plane* with an *inclined slope*. Without fine-tuning, our PhysAtObj predicts physically plausible future rollouts across both configurations, confirming that the interaction priors learn inherent physical properties. More details are in Appendix A.10.

Generalization to Multiple Scenes. To evaluate whether the module can learn different dynamic interactions across multiple 3D scenes, we train a single unified interaction module across all scenes in the Moving-Static Dataset simultaneously. Particularly, canonical representations and deformation fields adapt to individual scenes, while the shared interaction module updates across all scenes. As shown in Table 1, our method, denoted as *PhysAtObj*$_{multiScenes}$, achieves performance on par with per-scene models, showing that our module can learn general physical priors across multiple scenes.

Due to limited space, our additional sim-to-real experiments and analysis of the learned velocity changes are provided in Appendix A.11 and A.12.

4.3 Ablation Study

We conduct extensive ablation experiments on the Moving-Static Object Interaction Dataset.

(1) $\sim$ (4) Removing f_{field} / f_{normal} / $f_{friction}$: This set of experiments aims to evaluate how each sub-network within our core object interaction module models impulse-induced velocity changes.

(5) $\sim$ (7) Removing object-level interaction information: This series of experiments investigates the necessity of object-level information in learning accurate interaction dynamics. In setting (5), we remove object-level awareness entirely: the three networks f_{field} / f_{normal} / $f_{friction}$ operate directly on individual Gaussians $\boldsymbol{p}_t$ rather than object codes $\boldsymbol{o}_k$ or $\boldsymbol{o}_{con}$, and do not utilize the contact region direction $\boldsymbol{u}$. Consequently, the networks must predict the velocity changes $\delta\mathbb{V}_f/\boldsymbol{j}^n_{\boldsymbol{p}_c}/\boldsymbol{j}^r_{\boldsymbol{p}_c}$ at a strictly localized, per-Gaussian level. Settings (6) and (7) gradually add object codes back.

Table 2 shows the ablation results for future frame extrapolation. We can see that: 1) Removing one or more velocity change networks leads to a clear drop in both object segmentation accuracy and visual quality. This decline is particularly severe when omitting f_{field} which typically captures the velocity change induced by constant gravity, validating the necessity of learning comprehensive object dynamics. 2) Relying solely on point-level data to model these dynamics results in a catastrophic failure, underscoring the critical need to integrate object-level context when learning interactions.

(8) Removing f_{prop}: We conduct this ablation on a key PhysInOne deformable 3D scene involving a falling elastic cube. Without f_{prop}, the future dynamics of the elastic cube can hardly be predicted. More results are in Appendix A.13.

Table 2: Quantitative results of ablation studies.

	f_{field}	f_{normal}	$f_{friction}$	AP↑	PQ↑	F1↑	Pre↑	Rec↑	mIoU↑	mPSNR↑	mSSIM↑	mLPIPS↓	PSNR↑	SSIM↑	LPIPS↓
(1)	✓	✗	✗	46.023	39.843	55.042	55.473	55.018	47.483	18.790	0.714	0.553	30.280	**0.888**	0.164
(2)	✓	✓	✗	50.838	43.755	60.890	64.560	60.605	52.415	19.171	0.722	0.517	30.355	0.883	0.170
(3)	✓	✗	✓	44.413	38.088	53.875	54.290	53.890	46.020	18.567	0.712	0.501	29.974	0.867	0.188
(4)	✗	✓	✓	34.720	26.648	41.518	34.386	41.437	35.393	17.257	0.679	0.634	29.065	0.880	0.177
	$\boldsymbol{o}_k$	$\boldsymbol{o}_{con}$	$\boldsymbol{u}$	AP↑	PQ↑	F1↑	Pre↑	Rec↑	mIoU↑	mPSNR↑	mSSIM↑	mLPIPS↓	PSNR↑	SSIM↑	LPIPS↓
(5)	✗	✗	✗	31.160	26.390	39.215	37.618	43.933	32.138	17.106	0.677	0.587	28.054	0.875	0.184
(6)	✓	✗	✗	43.423	36.963	51.708	50.075	51.403	44.445	18.386	0.704	0.572	30.024	0.882	0.171
(7)	✓	✓	✗	37.245	31.188	45.798	46.380	45.618	38.590	17.587	0.680	0.609	29.373	0.881	0.174
PhysAtObj				**66.760**	**59.615**	**77.278**	**77.648**	**77.140**	**67.255**	**21.344**	**0.757**	**0.385**	**31.475**	0.884	**0.163**

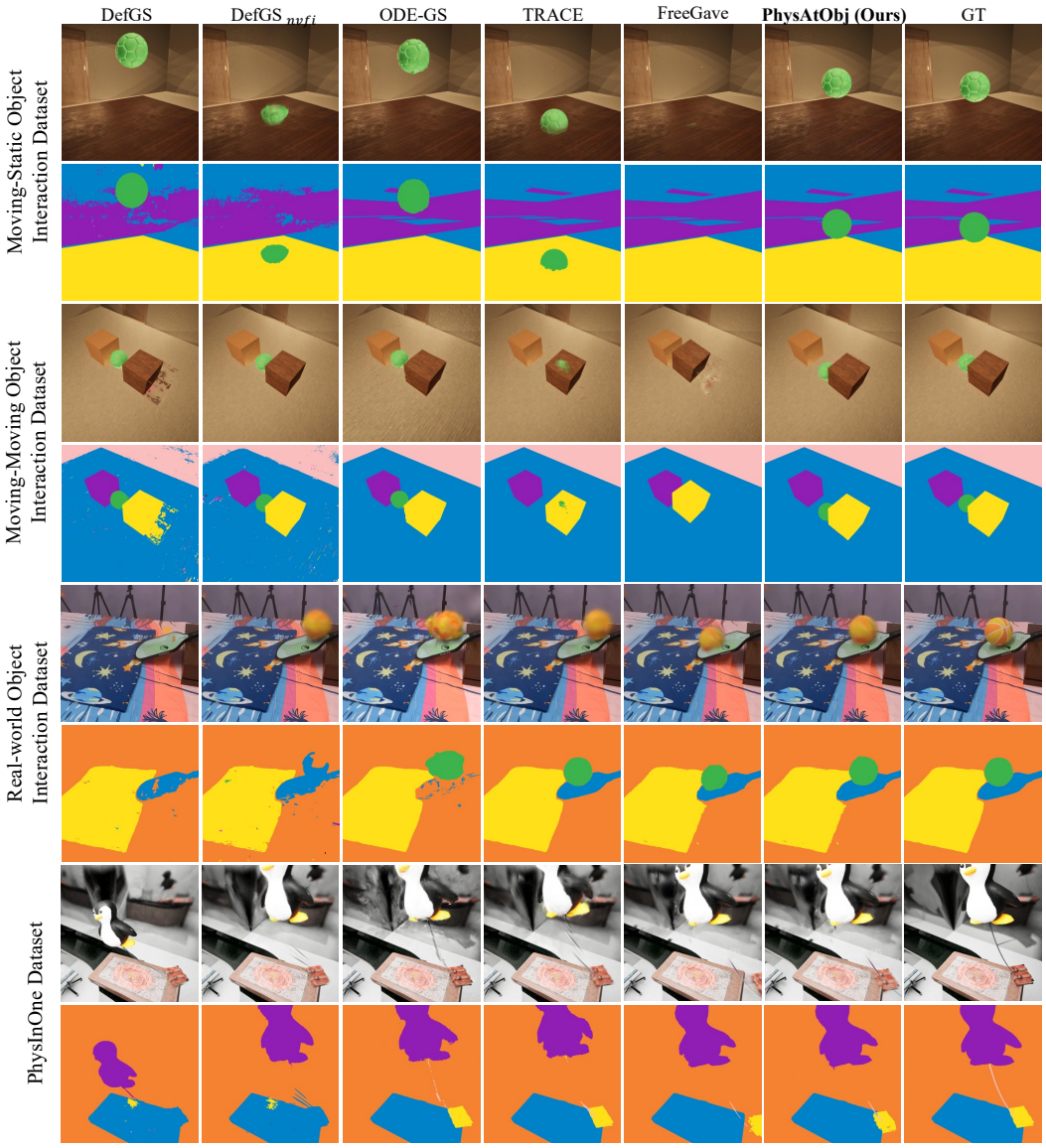

Figure 4: Qualitative results of future frame extrapolation on four datasets.



Figure 5: Generalization to different initial velocities and different geometries.

5 Conclusion

We present PhysAtObj, a new approach that models dynamic 3D scenes and predicts future frames directly from multi-view videos. Rather than merely interpolating observed motions or relying on predefined physical properties like mass and materials, our method explicitly learns underlying physical interactions. Using a specialized object interaction module, we systematically model time-agnostic, impulse-induced velocity changes during interactions. Extensive experiments on synthetic and real-world datasets show that PhysAtObj achieves state-of-the-art results, successfully generalizing learned dynamics to future frames.

AI Use Statement: In this work, we used generative AI tools to assist with code debugging during the development of the proposed method. We did not use generative AI tools to generate synthetic datasets, formulate mathematical claims or proofs, or perform qualitative or thematic data analysis. Generative AI tools were also used for improving the readability and clarity of the manuscript.

Reproducibility Statement: We provide the implementation and experimental details necessary to reproduce our results. All network architectures, datasets, training and testing details, and ablation studies are included in the main text and Appendix. We will release the source code, datasets, and trained models to facilitate reproduction of the reported results.

REFERENCES

Kelsey R. Allen, Yulia Rubanova, Tatiana Lopez-Guevara, William Whitney, Alvaro Sanchez-Gonzalez, Peter Battaglia, and Tobias Pfaff. Learning Rigid Dynamics With Face Interaction Graph Networks. *ICLR*, 2023.

Jeongmin Bae, Seoha Kim, Youngsik Yun, Hahyun Lee, Gun Bang, and Youngjung Uh. Per-Gaussian Embedding-Based Deformation for Deformable 3D Gaussian Splatting. *ECCV*, 2024.

Daniele Baieri, Stefano Esposito, Filippo Maggioli, and Emanuele Rodolà. Fluid Dynamics Network: Topology-Agnostic 4D Reconstruction via Fluid Dynamics Priors. *arXiv:2303.09871*, 2023.

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

Piotr Borycki, Weronika Smolak, Joanna Waczyńska, Marcin Mazur, Sławomir Tadeja, and Przemysław Spurek. GASP: Gaussian Splatting for Physic-Based Simulations. *arXiv:2409.05819*, 2024.

Junhao Cai, Yuji Yang, Weihao Yuan, Yisheng He, Zilong Dong, Liefeng Bo, Hui Cheng, and Qifeng Chen. Gaussian-Informed Continuum for Physical Property Identification and Simulation. *NeurIPS*, 2024.

Ang Cao and Justin Johnson. HexPlane: A Fast Representation for Dynamic Scenes. *CVPR*, 2023.

Junyi Cao, Shanyan Guan, Yanhao Ge, Wei Li, Xiaokang Yang, and Chao Ma. NeuMA: Neural Material Adaptor for Visual Grounding of Intrinsic Dynamics. *NeurIPS*, 2024.

Boyuan Chen, Hanxiao Jiang, Shaowei Liu, Saurabh Gupta, Yunzhu Li, Hao Zhao, and Shenlong Wang. PhysGen3D: Crafting a Miniature Interactive World from a Single Image. *CVPR*, 2025a.

Jianing Chen, Zehao Li, Yujun Cai, Hao Jiang, Chengxuan Qian, Juyuan Kang, Shuqin Gao, Honglong Zhao, Tianlu Mao, and Yucheng Zhang. HAIF-GS: Hierarchical and Induced Flow-Guided Gaussian Splatting for Dynamic Scene. *NeurIPS*, 2025b.

Xinyuan Chen, Yaohui Wang, Lingjun Zhang, Shaobin Zhuang, Xin Ma, Jiashuo Yu, Yali Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. SEINE: Short-to-Long Video Diffusion Model for Generative Transition and Prediction. *ICLR*, 2024.

Woojin Cho, Minju Jo, Haksoo Lim, Kookjin Lee, Dongeun Lee, Sanghyun Hong, and Noseong Park. Parameterized Physics-informed Neural Networks for Parameterized PDEs. *ICML*, 2024.

Mengyu Chu, Lingjie Liu, Quan Zheng, Erik Franz, Hans Peter Seidel, Christian Theobalt, and Rhaleb Zayer. Physics informed neural fields for smoke reconstruction with sparse data. *TOG*, 2022.

Danny Driess, Zhiao Huang, Yunzhu Li, Russ Tedrake, and Marc Toussaint. Learning Multi-Object Dynamics with Compositional Neural Radiance Fields. *CoRL*, 2022.

Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B Tenenbaum, and Jiajun Wu. Neural Radiance Flow for 4D View Synthesis and Video Processing. *ICCV*, 2021.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling Rectified Flow Transformers for High-Resolution Image Synthesis. *ICML*, 2024.

Jiemin Fang, Xinggang Wang, and Matthias Nießner. Fast Dynamic Radiance Fields with Time-Aware Neural Voxels. *SIGGRAPH Asia*, 2022.

Chun Feng, Hao Zhang, Haolan Xu, and Narendra Ahuja. Seeing is Simulating: Differentiable Physics for Interaction-Aware Material Estimation. *TMLR*, 2026.

Yutao Feng, Xiang Feng, Yintong Shang, Ying Jiang, Chang Yu, Zeshun Zong, Tianjia Shao, Hongzhi Wu, Kun Zhou, Chenfanfu Jiang, and Yin Yang. Gaussian Splashing: Dynamic Fluid Synthesis with Gaussian Splatting. *arXiv:2401.15318*, 2024a.

Yutao Feng, Yintong Shang, Xuan Li, Tianjia Shao, Chenfanfu Jiang, Yin Yang, and Los Angeles. PIE-NeRF: Physics-based Interactive Elastodynamics with NeRF. *CVPR*, 2024b.

Lin Geng Foo, Mark He Huang, Alexandros Lattas, Stylianos Moschoglou, Thabo Beeler, and Christian Theobalt. Physical Simulator In-the-Loop Video Generation. *CVPR*, 2026.

Sara Fridovich-Keil, Giacomo Meanti, Frederik Warburg, Benjamin Recht, and Angjoo Kanazawa. K-Planes: Explicit Radiance Fields in Space, Time, and Appearance. *CVPR*, 2023.

Zhoujie Fu, Jiacheng Wei, Wenhao Shen, Chaoyue Song, Xiaofeng Yang, Fayao Liu, Xulei Yang, and Guosheng Lin. Sync4D: Video Guided Controllable Dynamics for Physics-Based 4D Generation. *arXiv:2405.16849*, 2024.

Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic View Synthesis from Dynamic Monocular Video. *ICCV*, 2021.

Zhiyuan Gao, Jiageng Mao, Hong-Xing Yu, Haozhe Lou, Emily Yue-Ting Jia, Jernej Barbic, Jiajun Wu, and Yue Wang. Seeing the Wind from a Falling Leaf. *NeurIPS*, 2025.

Zhongkai Hao, Chengyang Ying, Hang Su, Jun Zhu, Jian Song, and Ze Cheng. Bi-level Physics-Informed Neural Networks for PDE Constrained Optimization Using Broyden's Hypergrandients. *ICLR*, 2023.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J. Fleet. Video Diffusion Models. *NeurIPS*, 2022.

Tobias Höppe, Arash Mehrjou, Stefan Bauer, Didrik Nielsen, and Andrea Dittadi. Diffusion Models for Video Prediction and Infilling. *TMLR*, 2022.

Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. A Moving Least Squares Material Point Method with Displacement Discontinuity and Two-Way Rigid Body Coupling. *SIGGRAPH*, 2018.

Bei Huang, Yixin Chen, Ruijie Lu, Gang Zeng, Hongbin Zha, Yuru Pei, and Siyuan Huang. GaussianFluent: Gaussian Simulation for Dynamic Scenes with Mixed Materials. *arXiv:2601.09265*, 2026a.

Qilin Huang, Quynh Anh Huynh, Long Le, Chen Wang, Chuhao Chen, Ryan Lucas, Eric Eaton, and Lingjie Liu. UniPixie: Unified and Probabilistic 3D Physics Learning via Flow Matching. *CVPR*, 2026b.

Tianyu Huang, Yihan Zeng, Hui Li, Wangmeng Zuo, and Rynson W. H. Lau. DreamPhysics: Learning Physical Properties of Dynamic 3D Gaussians with Video Diffusion Priors. *AAAI*, 2025.

Woobin Im, Geonho Cha, Sebin Lee, Jumin Lee, Juhyeong Seon, Dongyoon Wee, and Sung-Eui Yoon. Regularizing Dynamic Radiance Fields with Kinematic Fields. *ECCV*, 2024.

Emily Yue-Ting Jia, Jiageng Mao, Zhiyuan Gao, Yajie Zhao, and Yue Wang. Learning an Implicit Physics Model for Image-based Fluid Simulation. *ICCV*, 2025.

Hanxiao Jiang, Hao-Yu Hsu, Kaifeng Zhang, Hsin-Ni Yu, Shenlong Wang, and Yunzhu Li. PhysTwin: Physics-Informed Reconstruction and Simulation of Deformable Objects from Videos. *ICCV*, 2025.

Alvaro Sanchez-gonzalez Jonathan, Godwin Tobias, Pfaff Rex, Ying Jure, and Peter W Battaglia. Learning to Simulate Complex Physics with Graph Networks. *ICML*, 2020.

Takuhiro Kaneko. Improving Physics-Augmented Continuum Neural Radiance Field-Based Geometry-Agnostic System Identification with Lagrangian Particle Optimization. *CVPR*, 2024.

Bingyi Kang, Yang Yue, Rui Lu, Zhijie Lin, Yang Zhao, Kaixin Wang, Gao Huang, and Jiashi Feng. How Far is Video Generation from World Model: A Physical Law Perspective. *ICML*, 2025.

Bernhard Kerbl, Université Côte, Georgios Kopanas, Université Côte, Thomas Leimkühler, Max-planck-institut Informatik, and G R Aug. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *SIGGRAPH*, 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:2508.17437*, 2025.

Jiahui Lei, Yijia Weng, Adam Harley, Leonidas Guibas, and Kostas Daniilidis. MoSca: Dynamic Gaussian Fusion from Casual Videos via 4D Motion Scaffolds. *CVPR*, 2025.

Yuanhang Lei, Boming Zhao, Zesong Yang, Xingxuan Li, Tao Cheng, Haocheng Peng, Ru Zhang, Yang Yang, Siyuan Huang, Yujun Shen, Ruizhen Hu, Hujun Bao, and Zhaopeng Cui. DiffWind: Physics-Informed Differentiable Modeling of Wind-Driven Object Dynamics. *ICLR*, 2026.

Jin Li, Yang Gao, Wenfeng Song, Yacong Li, Shuai Li, Aimin Hao, and Hong Qin. CoupNeRF: Property-aware Neural Radiance Fields for Multi-Material Coupled Scenario Reconstruction. *Computer Graphics Forum*, 2024a.

Jinxi Li, Ziyang Song, and Bo Yang. NVFi: Neural Velocity Fields for 3D Physics Learning from Dynamic Videos. *NeurIPS*, 2023a.

Jinxi Li, Ziyang Song, and Bo Yang. TRACE: Learning 3D Gaussian Physical Dynamics from Multi-view Videos. *ICCV*, 2025a.

Jinxi Li, Ziyang Song, Siyuan Zhou, and Bo Yang. FreeGave: 3D Physics Learning from Dynamic Videos by Gaussian Velocity. *CVPR*, 2025b.

Shiqian Li, Ruihong Shen, Junfeng Ni, Chang Pan, Chi Zhang, and Yixin Zhu. Learning Physics-Grounded 4D Dynamics with Neural Gaussian Force Fields. *ICLR*, 2026a.

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. *ICLR*, 2023b.

Yifei Li, Haixu Wu, Zeyi Xu, Tuur Stuyck, and Wojciech Matusik. Neural Modular Physics for Elastic Simulation. *ICML*, 2026b.

Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime Gaussian Feature Splatting for Real-Time Dynamic View Synthesis. *CVPR*, 2024b.

Junbang Liang, Ming C. Lin, and Vladlen Koltun. Differentiable cloth simulation for inverse problems. *NeurIPS*, 2019.

Chenguo Lin, Yuchen Lin, Panwang Pan, Yifan Yu, Honglei Yan, Katerina Fragkiadaki, and Yadong Mu. MoVieS: Motion-Aware 4D Dynamic View Synthesis in One Second. *CVPR*, 2026.

Youtian Lin, Zuozhuo Dai, Siyu Zhu, and Yao Yao. Gaussian-Flow: 4D Reconstruction with Dynamic 3D Gaussian Particle. *CVPR*, 2024.

Yuchen Lin, Chenguo Lin, Jianjin Xu, and Yadong Mu. OmniPhysGS: 3D Constitutive Gaussians for General Physics-Based Dynamics Generation. *ICLR*, 2025.

Yaron Lipman, Ricky T Q Chen, Heli Ben-hamu, Maximilian Nickel, and Matt Le. Flow Matching for Generative Modeling. *ICLR*, 2023.

Chunjiang Liu, Xiaoyuan Wang, Qingran Lin, Albert Xiao, Haoyu Chen, Shizheng Wen, Hao Zhang, Lu Qi, Ming-Hsuan Yang, Laszlo A. Jeni, Min Xu, and Yizhou Zhao. MOSIV: Multi-Object System Identification from Videos. *ICLR*, 2026.

Fangfu Liu, Hanyang Wang, Shunyu Yao, and Shengjun Zhang. Physics3D: Learning Physical Properties of 3D Gaussians via Video Diffusion. *arXiv:2406.04338*, 2024a.

Jia-Wei Liu, Yan-Pei Cao, Weijia Mao, Wenqiao Zhang, David Junhao Zhang, Jussi Keppo, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. DeVRF: Fast Deformable Voxel Radiance Fields for Dynamic Scenes. *NeurIPS*, 2022.

Qingming Liu, Yuan Liu, Jiepeng Wang, Xianqiang Lyv, Peng Wang, Wenping Wang, and Junhui Hou. MoDGS: Dynamic Gaussian Splatting from Casually-captured Monocular Videos with Depth Priors. *ICLR*, 2025.

Shaowei Liu, Zhongzheng Ren, Saurabh Gupta, and Shenlong Wang. PhysGen: Rigid-Body Physics-Grounded Image-to-Video Generation. *ECCV*, 2024b.

Yu-Lun Liu, Chen Gao, Andreas Meuleman, Hung-Yu Tseng, Ayush Saraf, Changil Kim, Yung-Yu Chuang, Johannes Kopf, and Jia-Bin Huang. Robust Dynamic Radiance Fields. *CVPR*, 2023.

Zhicheng Lu, Xiang Guo, Le Hui, Min Yang, Xiao Tang, Feng Zhu, and Yuchao Dai. 3D Geometry-aware Deformable Gaussian Splatting for Dynamic View Synthesis. *CVPR*, 2024.

Chunji Lv, Zequn Chen, Donglin Di, Weinan Zhang, Hao Li, Wei Chen, and Changsheng Li. PhysGM: Large Physical Gaussian Model for Feed-Forward 4D Synthesis. *CVPR*, 2026.

Marko Mihajlovic, Sergey Prokudin, Siyu Tang, Robert Maier, Federica Bogo, Tony Tung, and Edmond Boyer. SplatFields: Neural Gaussian Splats for Sparse 3D and 4D Reconstruction. *ECCV*, 2024.

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. *ECCV*, 2020.

Saman Motamed, Laura Culp, Kevin Swersky, Priyank Jaini, and Robert Geirhos. Do generative video models learn physical principles from watching videos? *arXiv:2501.09038*, 2025.

Haomiao Ni, Changhao Shi, Kai Li, Sharon X. Huang, and Martin Renqiang Min. Conditional Image-to-Video Generation with Latent Flow Diffusion Models. *CVPR*, 2023.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. *CVPR*, 2019.

Jongmin Park, Minh-Quan Viet Bui, Juan Luis Gonzalez Bello, Jaeho Moon, Jihyong Oh, and Munchurl Kim. SplineGS: Robust Motion-Adaptive Spline for Real-Time Dynamic 3D Gaussians from Monocular Video. *CVPR*, 2025.

Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. HyperNeRF: A Higher-Dimensional Representation for Topologically Varying Neural Radiance Fields. *SIGGRAPH Asia*, 2021.

Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. *CVPR*, 2021.

Zhiyin Qian, Shaofei Wang, Marko Mihajlovic, Andreas Geiger, and Siyu Tang. 3DGS-Avatar: Animatable Avatars via Deformable 3D Gaussian Splatting. *CVPR*, 2024.

Ri-Zhao Qiu, Ge Yang, Weijia Zeng, and Xiaolong Wang. Feature Splatting: Language-Driven Physics-Based Scene Synthesis and Editing. *ECCV*, 2024.

Jinsheng Quan, Qiaowei Miao, Yichao Xu, Zizhuo Lin, Ying Li, Wei Yang, Zhihui Li, and Yawei Luo. ParticleGS: Learning Neural Gaussian Particle Dynamics from Videos for Prior-free Physical Motion Extrapolation. *CVPR*, 2026.

Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 2020.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. SAM 2: Segment Anything in Images and Videos. *arXiv:2408.00714*, 2024.

Yulia Rubanova, Tatiana Lopez-Guevara, Kelsey R. Allen, William F. Whitney, Kimberly Stachenfeld, and Tobias Pfaff. Learning rigid-body simulators over implicit shapes for large-scale scenes and vision. *NeurIPS*, 2024.

Haochen Shi, Huazhe Xu, Zhiao Huang, Yunzhu Li, and Jiajun Wu. RoboCraft: Learning to see, simulate, and shape elasto-plastic objects in 3D with graph networks. *RSS*, 2024.

Xiyang Tan, Ying Jiang, Xuan Li, Zeshun Zong, Tianyi Xie, Yin Yang, and Chenfanfu Jiang. PhysMotion: Physics-Grounded Dynamics From a Single Image. *arXiv:2411.17189*, 2024.

Fengrui Tian, Shaoyi Du, and Yueqi Duan. MonoNeRF: Learning a Generalizable Dynamic Radiance Field from Monocular Videos. *ICCV*, 2023.

Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-Rigid Neural Radiance Fields: Reconstruction and Novel View Synthesis of a Dynamic Scene from Monocular Video. *ICCV*, 2021.

Federico Vasile, Ri-zhao Qiu Lorenzo, and Natale Xiaolong. Gaussian-Augmented Physics Simulation and System Identification with Complex Colliders. *NeurIPS*, 2025.

Vikram Voleti, Alexia Jolicoeur-Martineau, and Christopher Pal. MCVD: Masked Conditional Video Diffusion for Prediction, Generation, and Interpolation. *NeurIPS*, 2022.

Team Wan. Wan: Open and Advanced Large-Scale Video Generative Models. *arXiv:2503.20314*, 2025.

Boyuan Wang, Xiaofeng Wang, Yongkang Li, Zheng Zhu, Yifan Chang, Angen Ye, Guosheng Zhao, Chaojun Ni, Guan Huang, Yijie Ren, Yueqi Duan, and Xingang Wang. ReconPhys: Reconstruct Appearance and Physical Attributes from Single Video. *ECCV*, 2026a.

Dan Wang, Xinrui Cui, Serge Belongie, and Ravi Ramamoorthi. PhysConvex: Physics-Informed Dynamic Convex Fields for Reconstruction and Simulation. *ECCV*, 2026b.

Daniel Wang, Patrick Rim, Tian Tian, Alex Wong, and Ganesh Sundaramoorthi. ODE-GS: Latent ODEs for Dynamic Scene Extrapolation with 3D Gaussian Splatting. *ICLR*, 2026c.

Xiaogang Wang, Hongyu Wu, Wenfeng Song, and Kai Xu. Fracture-GS: Dynamic Fracture Simulation with Physics-Integrated Gaussian Splatting. *ICLR*, 2026d.

William F. Whitney, Tatiana Lopez-Guevara, Tobias Pfaff, Yulia Rubanova, Thomas Kipf, Kimberly Stachenfeld, and Kelsey R. Allen. Learning 3D Particle-based Simulators from RGB-D Videos. *ICLR*, 2024.

Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4D Gaussian Splatting for Real-Time Dynamic Scene Rendering. *CVPR*, 2024.

Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. PhysGaussian: Physics-Integrated 3D Gaussians for Generative Dynamics. *CVPR*, 2024.

Zhen Xu, Zhengqin Li, Zhao Dong, Xiaowei Zhou, Richard Newcombe, and Zhaoyang Lv. 4DGT: Learning a 4D Gaussian Transformer Using Real-World Monocular Videos. *NeurIPS*, 2025.

Haotian Xue, Antonio Torralba, Joshua B. Tenenbaum, Daniel LK Yamins, Yunzhu Li, and Hsiao-Yu Tung. 3D-IntPhys: Towards More Generalized 3D-grounded Visual Intuitive Physics under Challenging Scenes. *CVPR Workshops*, 2023.

Jing-wen Yang, Jia-mu Sun, Yong-liang Yang, and Jie Yang. DMiT: Deformable Mipmapped Tri-Plane Representation for Dynamic Scenes. *ECCV*, 2024a.

Zeyu Yang, Hongye Yang, Zijie Pan, Xiatian Zhu, and Li Zhang. Real-time Photorealistic Dynamic Scene Representation and Rendering with 4D Gaussian Splatting. *ICLR*, 2024b.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Xiaotao Gu, Yuxiao Dong, and Jie Tang. CogVideoX: Text-to-Video Diffusion Mod- els with An Expert Transformer. *ICLR*, 2025.

Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3D Gaussians for High-Fidelity Monocular Dynamic Scene Reconstruction. *CVPR*, 2024c.

Yuxin Yao, Siyu Ren, Junhui Hou, Zhi Deng, Juyong Zhang, and Wenping Wang. DynoSurf: Neural Deformation-based Temporally Consistent Dynamic Surface Reconstruction. *ECCV*, 2024.

Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian Grouping: Segment and Edit Anything in 3D Scenes. *ECCV*, 2024.

Suwoong Yeom, Joonsik Nam, Seunggyu Choi, Lucas Yunkyu Lee, Sangmin Kim, Jaesik Park, Joonsoo Kim, Kugjin Yun, Kyeongbo Kong, and Suk-Ju Kang. TRiGS: Temporal Rigid-Body Motion for Scalable 4D Gaussian Splatting. *ECCV*, 2026.

Lijun Yu, Yong Cheng, Kihyuk Sohn, José Lezama, Han Zhang, Huiwen Chang, Alexander G. Hauptmann, Ming-Hsuan Yang, Yuan Hao, Irfan Essa, and Lu Jiang. MAGVIT: Masked Generative Video Transformer. *CVPR*, 2023.

Yifan Zhan, Zhuoxiao Li, Muyao Niu, Zhihang Zhong, Shohei Nobuhara, Ko Nishino, and Yinqiang Zheng. KFD-NeRF: Rethinking Dynamic NeRF with Kalman Filter. *ECCV*, 2024.

Kaifeng Zhang, Baoyu Li, Kris Hauser, and Yunzhu Li. Particle-Grid Neural Dynamics for Learning Deformable Object Models from RGB-D Videos. *RSS*, 2025.

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. *ECCV*, 2024.

Mikel Zhobro, Andreas René Geist, and Georg Martius. Learning 3D-Gaussian Simulators from RGB Videos. *arXiv:2503.24009*, 2025.

Licheng Zhong, Hong-xing Yu, Jiajun Wu, and Yunzhu Li. Reconstruction and Simulation of Elastic Objects with Spring-Mass 3D Gaussians. *ECCV*, 2024.

Siyuan Zhou, Hejun Wang, Hu Cheng, Jinxi Li, Dongsheng Wang, Junwei Jiang, Yixiao Jin, Jiayue Huang, Shiwei Mao, Shangjia Liu, Yafei Yang, Hongkang Song, Shenxing Wei, Zihui Zhang, Peng Huang, Shijie Liu, Zhengli Hao, Hao Li, Yitian Li, Wenqi Zhou, Zhihan Zhao, Zongqi He, Hongtao Wen, Shouwang Huang, Peng Yun, Bowen Cheng, Pok Kazaf Fu, Wai Kit Lai, Jiahao Chen, Kaiyuan Wang, Zhixuan Sun, Ziqi Li, Haochen Hu, Di Zhang, Chun Ho Yuen, Bing Wang, Zhihua Wang, Chuhang Zou, and Bo Yang. Physinone: Visual physics learning and reasoning in one suite. *CVPR*, 2026.

A APPENDIX

A.1 MORE DETAILS OF OBJECT CONTACT REGION AND DIRECTION

contact region extraction: As described in Sec 3.3, for each Gaussians $\boldsymbol{p}_t$ in object $\boldsymbol{o}_k$, we search for the nearest Gaussian $\boldsymbol{p}_{near}$ that belong to another object $\boldsymbol{o}_j$, $k \neq j$. If the Euclidean distance between $\boldsymbol{p}_t$ and $\boldsymbol{p}_{near}$ is within a threshold δd, $\boldsymbol{p}_t$ is contacted with object $\boldsymbol{o}_j$ on $\boldsymbol{p}_{near}$. For object $\boldsymbol{o}_k$, if multiple of its Gaussians contact the same $\boldsymbol{o}_j$, they form a contact region, representing the location where $\boldsymbol{o}_k$ is contacted with $\boldsymbol{o}_j$.

contact direction estimation: For each contact region, we determine its surface-normal-aligned contact direction as follows:

- If the region contains more than 3 Gaussians, we use PCA (Principle Component Analysis) to compute the surface normal. We first compute the centered point coordinates of the region. Then we estimate the covariance matrix and perform eigen-decomposition. The eigenvector corresponding to the smallest eigenvalue defines the surface normal of the object.
- To determine the direction of the surface normal, we denote the contact-region center on object $\boldsymbol{o}_k$ by $\bar{\boldsymbol{p}}_{\mathcal{C},k}$ and the center of its matched points on object $\boldsymbol{o}_j$ by $\bar{\boldsymbol{p}}_{\mathcal{C},j}$. We orient the normal along $\bar{\boldsymbol{p}}_{\mathcal{C},k} - \bar{\boldsymbol{p}}_{\mathcal{C},j}$, toward $\boldsymbol{o}_k$.
- If the region contains fewer than 3 Gaussians, we use the unit direction of $\bar{\boldsymbol{p}}_{\mathcal{C},k} - \bar{\boldsymbol{p}}_{\mathcal{C},j}$ as the contact direction.

A.2 DETAILS OF STATE-AWARE VELOCITY PROPAGATION

We implement f_{prop} with a neighborhood aggregation MLP and a shared prediction head. The following steps map the contact responses and object state to the rigid and local changes in Equation 9.

1) Align the object state. For Gaussian i of object $\boldsymbol{o}_k$, let $\boldsymbol{p}_i^0$, $\boldsymbol{p}_i$, and $\boldsymbol{v}_i^{local}$ denote its canonical position, current position, and local velocity. Let $\bar{\boldsymbol{p}}^0$ and $\bar{\boldsymbol{p}}$ be the object centers, and $\mathbf{Q}_k$ the best-fit rotation from the centered canonical shape to the current shape. We define canonical positions $\boldsymbol{x}_i^0$, aligned positions $\boldsymbol{x}_i$, local velocities $\boldsymbol{z}_i$, and shape displacements $\boldsymbol{e}_i$ by

$$\begin{aligned} \boldsymbol{x}_i^0 &= \boldsymbol{p}_i^0 - \bar{\boldsymbol{p}}^0, \quad & \boldsymbol{x}_i &= \mathbf{Q}_k^T(\boldsymbol{p}_i - \bar{\boldsymbol{p}}), \\ \boldsymbol{z}_i &= \mathbf{Q}_k^T \boldsymbol{v}_i^{local}, \quad & \boldsymbol{e}_i &= \boldsymbol{x}_i - \boldsymbol{x}_i^0. \end{aligned} \tag{11}$$

Contact normals and responses, field responses, and rigid velocities are rotated into the same frame.

2) Aggregate neighborhood features. Let $\mathcal{N}_i$ contain the 12 nearest same-object neighbors in the canonical shape. The point feature $\boldsymbol{a}_i$ collects contact occupancy, aligned normal and response, their position–response cross product, $\boldsymbol{x}_i$, $\boldsymbol{z}_i$, $\boldsymbol{e}_i$, field response, and rigid velocity. Let $\boldsymbol{g}_k$ be its object summary, and $\boldsymbol{b}_{ij}$ collect the transformed edge-length ratio, edge direction, relative local velocity, and radial relative velocity. With zero self-edge feature $\boldsymbol{b}_{ii}$, aggregation MLP f_{agg}, and neighborhood feature $\boldsymbol{h}_i$, we compute

$$\boldsymbol{h}_i = \frac{1}{1 + |\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i \cup \{i\}} f_{agg}(\boldsymbol{a}_i, \boldsymbol{a}_j, \boldsymbol{g}_k, \boldsymbol{b}_{ij}). \tag{12}$$

The summary averages over the object; for the rigid readout, its contact-normal, response, and cross-product entries instead average over contact points. This readout reuses the same MLPs.

3) Decode rigid and local responses. Let $\boldsymbol{c}_k$ be a learned physical embedding, $\boldsymbol{s}_k$ the aligned center and angular velocities, and δt_0 a base time step. The shared head f_{head} produces six rigid coefficients $\boldsymbol{g}_i^{rig}$, three local-contact coefficients $\boldsymbol{g}_i^{loc}$, and two rate logits $\boldsymbol{g}_i^{rate}$:

$$(\boldsymbol{g}_i^{rig}, \boldsymbol{g}_i^{loc}, \boldsymbol{g}_i^{rate}) = f_{head}\big(\boldsymbol{h}_i, \boldsymbol{o}_k, \boldsymbol{c}_k, \boldsymbol{x}_i^0, \boldsymbol{x}_i, \boldsymbol{z}_i, \boldsymbol{e}_i, \boldsymbol{s}_k, \delta t/\delta t_0\big)\,. \tag{13}$$

We apply a scaled $\tanh$ to the rigid and local-contact coefficients. Contact-point averaging of the rigid readout yields the object-level translation and rotation increments; rotating them back and converting to the basis of Equation 4 gives $\delta\mathbb{V}_{\boldsymbol{o}_k}^{con}$. The local-contact output gives a per-point increment $\boldsymbol{d}_i^{con}$ in the canonical frame. Both contact outputs vanish when the object has no contact, while $\delta\mathbb{V}_f$ remains active.

4) Maintain deformation without adding rigid motion. We reuse the same MLPs with only canonical geometry and object codes to decode positive, bounded restoring and damping rates. Let $\boldsymbol{a}_i^{rest}$ and $\boldsymbol{a}_i^{damp}$ denote their velocity-rate responses: restoring acts on edge-length deviations from the canonical shape, and damping opposes relative local velocity along each edge. Edge responses are equal and opposite, with an additional weak canonical-shape restoring term. Let Π_k^{local} subtract the equal-weight least-squares rigid field from a point-vector field. Then

$$\delta \boldsymbol{v}_i^{local} = \left[\Pi_k^{local} \left(\left\{ \mathbf{Q}_k \left[\boldsymbol{d}_i^{con} + \delta t (\boldsymbol{a}_i^{rest} + \boldsymbol{a}_i^{damp}) \right] \right\}_{i \in \boldsymbol{o}_k} \right) \right]_i . \tag{14}$$

The outer subscript selects point i. Restoring and damping remain active after contact ends. Projection removes the rigid component of the local increment at the current geometry.

5) Bound stretching during integration. For canonical edge (i, j), let L_{ij}^{obs} be the maximum canonical/training-observed length, $L_{min} > 0$ a numerical floor, and η the expansion factor. Let $\boldsymbol{p}_i^+$ denote the next position. We constrain

$$\|\boldsymbol{p}_i^+ - \boldsymbol{p}_j^+\| \leq \max(\eta L_{ij}^{obs}, L_{min}), \qquad \eta = 1.2 \text{ for main experiments.} \tag{15}$$

We correct the candidate local displacement, remove induced rigid motion, and scale it if needed to satisfy the bounds within numerical tolerance. The local velocity follows the corrected displacement; predicted rigid motion is preserved. Thus, the bound applies during inference and is relative to observed edge lengths, not solely the canonical shape.

Implementation. The aggregation MLP has three linear layers of width 32; the shared head has two hidden layers of width 64 and 11 outputs. Both use SiLU between linear layers and fixed feature-normalization scales. All steps repeat during rollout using the updated state and fixed canonical geometry, without deformation-field corrections after initialization.

A.3 More Details of Learning Velocity Changes

- f_{field}: This network includes 6 MLP layers, and each hidden layer has a size of 128 neurons.
- f_{normal}: This network includes 6 MLP layers, and each hidden layer has a size of 128 neurons.
- $f_{friction}$: This network includes 6 MLP layers, and each hidden layer has a size of 128 neurons.
- f_{prop}: This module reuses a neighborhood aggregation MLP and a shared prediction head to predict rigid and local velocity changes; details are provided in Appendix A.2.

A.4 More Details of the Training Algorithm

Algorithm 1 Mid-point method.

Input: (1) Gaussians $\boldsymbol{P}_{t'}$ of a target object $\boldsymbol{o}_k$ at t': $\{\cdot\cdot(\boldsymbol{p}_{t'}, \boldsymbol{s}_{t'}, \boldsymbol{r}_{t'}, \boldsymbol{o}_k, \sigma, \boldsymbol{c})\cdot\cdot\}$; (2) δt;
(3) Shared rigid velocity $\mathbb{V}_{\boldsymbol{o}_k}$ and local velocity $\boldsymbol{v}_{\boldsymbol{p}_{t'}}^{local}$ at time t';
(4) Field velocity change $\delta\mathbb{V}_f$;
(5) Contact-induced rigid velocity change $\delta\mathbb{V}_{\boldsymbol{o}_k}^{con}$ and local velocity change $\delta\boldsymbol{v}_{\boldsymbol{p}_{t'}}^{local}$ over δt.

Output: Gaussians $\boldsymbol{P}_t$ at t, *i.e.*, at $(t' + \delta t)$: $\{\cdot\cdot(\boldsymbol{p}_t, \boldsymbol{s}_t, \boldsymbol{r}_t, \boldsymbol{o}_k, \sigma, \boldsymbol{c})\cdot\cdot\}$

- Calculate the mid-point velocity for a Gaussian $\boldsymbol{p}_{t'}$ moving from t' to t:
  $\boldsymbol{v}_{mid} \leftarrow \mathbb{V}_{\boldsymbol{o}_k} \cdot \mathcal{B}(\boldsymbol{p}_{t'}) + \boldsymbol{v}_{\boldsymbol{p}_{t'}}^{local} + \frac{1}{2}\Big((\delta\mathbb{V}_f + \delta\mathbb{V}_{\boldsymbol{o}_k}^{con}) \cdot \mathcal{B}(\boldsymbol{p}_{t'}) + \delta\boldsymbol{v}_{\boldsymbol{p}_{t'}}^{local}\Big)$;
- Calculate Gaussian $\boldsymbol{p}_t$ at time t: $\boldsymbol{p}_t \leftarrow \boldsymbol{p}_{t'} + \delta t \cdot \boldsymbol{v}_{mid}$;
- Calculate rotation $\boldsymbol{r}_t$ for the Gaussian $\boldsymbol{p}_t$ at time t:
  $\mathbf{R}_t \leftarrow (\mathbf{I} + \delta t \frac{\partial \boldsymbol{v}_{mid}}{\partial \boldsymbol{p}_{mid}})\mathbf{R}_{t'}$, where $\mathbf{R}_{t'}$ is converted from $\boldsymbol{r}_{t'}$ and $\boldsymbol{p}_{mid} \leftarrow \boldsymbol{p}_{t'} + \frac{\delta t}{2}\Big(\mathbb{V}_{\boldsymbol{o}_k} \cdot \mathcal{B}(\boldsymbol{p}_{t'}) + \boldsymbol{v}_{\boldsymbol{p}_{t'}}^{local}\Big)$, following the approximation in (Xie et al., 2024); Convert rotation back to quaternion: $\boldsymbol{r}_t \leftarrow \mathbf{R}_t$;
- The remaining Gaussian properties $\boldsymbol{s}_t/\boldsymbol{o}_k/\sigma/\boldsymbol{c}$ are held constant over time.

Return: Gaussians of object $\boldsymbol{o}_k$ at t: $\{\cdot\cdot(\boldsymbol{p}_t, \boldsymbol{r}_t, \boldsymbol{s}_t, \boldsymbol{o}_k, \sigma, \boldsymbol{c})\cdot\cdot\}$. All objects are updated simultaneously.

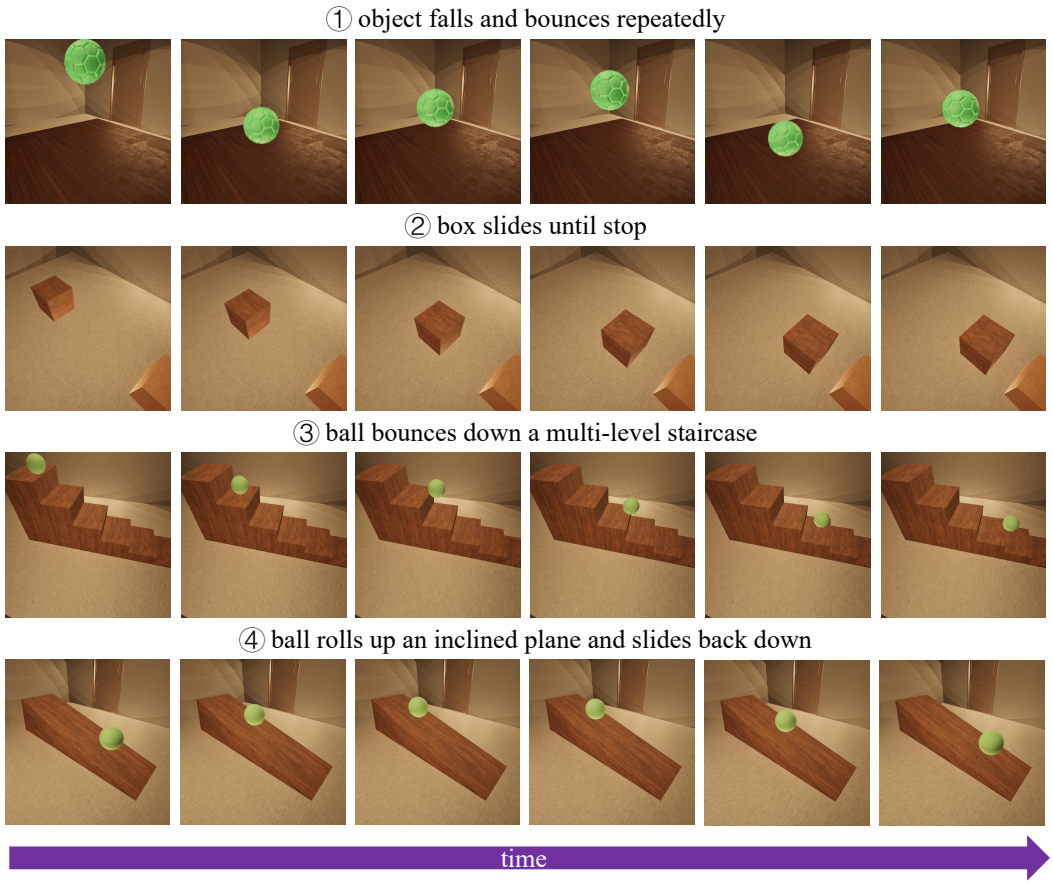

Figure 6: Qualitative examples from the Moving-Static Object Interaction Dataset.

A.5 More Details of Moving-Static Object Interaction Dataset

All scenes in this dataset are built with Unreal Engine. Specifically, we employ the *Chaos Physics* system in UE to simulate object dynamics.

For each 3D scene, we place 20 static cameras uniformly across the upper hemisphere at elevation angles $0^\circ \sim 60^\circ$ to capture comprehensive multi-view coverage. All videos are rendered at 1120×1120 resolution at 30 FPS.

Qualitative examples for Moving-Static Object Interaction Dataset are shown in Figure 6. Detailed dynamics for each scene are as follows:

① *object falls and bounces repeatedly*: The green ball falls freely from mid-air, accelerating downwards under gravity before hitting the ground. After colliding with the ground, it bounces upwards. Then, under gravity, it decelerates before falling freely again, repeating this process multiple times.

② *box slides until stop*: There is a stationary box and a sliding box on the ground. The sliding box slides towards the stationary box. Due to the friction of the ground, it gradually slows down until it comes to a stop without colliding with the stationary box.

③ *ball bounces down a multi-level staircase*: There is a multi-step staircase on the ground. A green ball rolls down the staircase from the top with a certain initial horizontal velocity. As it falls, it is affected by gravity and collides with the staircase, constantly changing its velocity and direction, and eventually bounces down the staircase.

④ *ball rolls up an inclined plane and slides back down*: There is a slope on the ground. A green ball rolls up the slope with a certain initial velocity. As it moves upward, its speed gradually decreases

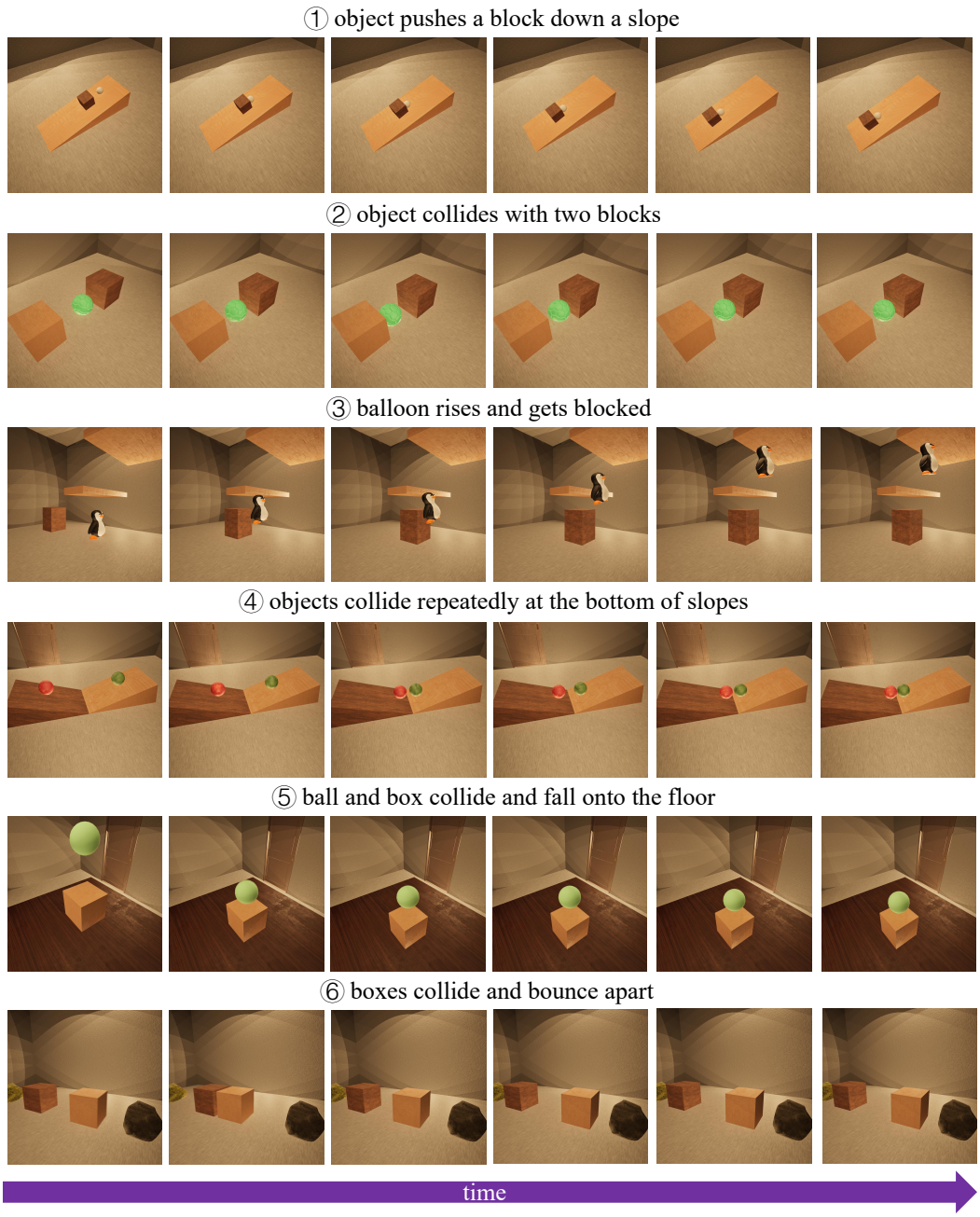

Figure 7: Qualitative examples from the Moving-Static Object Interaction Dataset.

until it comes to rest at the top of the slope. Then, under the combined force of gravity and the support force of the slope, it moves downward and rolls towards the bottom of the slope.

More dataset statistics are as follows:

- Number of scenes: 4.
- Average number of moving interaction objects: 1.
- Average number of static interaction objects: 1.5.
- Average video length: 3.75 seconds.
- Total number of cameras per scene: 20.
- Frames Per Second (FPS): 30.
- Number of training views: 17.
- Number of novel views: 3.

- Training time: first 50% of each video (involving object interactions).
- Testing time: remaining 50% of each video (involving object interactions).

A.6 More Details of Moving-Moving Object Interaction Dataset

All scenes in this dataset are built with Unreal Engine under the same simulator and camera settings as the Static-Moving Object Interaction Dataset. Qualitative examples for this dataset are provided in Figure 7. Detailed dynamics for each scene are as follows:

① *object pushes a block down a slope*: The ball slides down the slope from the top and collides with a wooden block that is slowly sliding down the slope. After the collision, the ball and the block move down the slope together with the same velocity.

② *object collides with two blocks*: There are two wooden blocks on the ground, one fixed and the other movable. A green ball rolls on the ground with a certain initial velocity and collides with the two blocks. When it hits the fixed block, the ball bounces back directly. When it hits the movable block, the block's velocity changes due to the impact.

③ *balloon rises and gets blocked*: Initially, both the balloon and the wooden block are on the ground. The wooden block slides towards the balloon with an initial velocity, causing the balloon to float upwards due to buoyancy until it is stopped by a wooden plank suspended in mid-air. The wooden block then collides with the balloon, causing it to leave the mid-air barrier and continue floating until it collides with another identical wooden plank in mid-air and comes to a stop. The moving wooden block eventually also comes to a stop.

④ *objects collide repeatedly at the bottom of slopes*: There are two inclined planes on the ground, with a small ball placed at the top of each plane. Under the influence of gravity and the supporting force of the ground, the two balls move downwards along the inclined planes. They collide at the bottom of the inclined planes, bounce off, and then slide down the planes again, colliding again at the bottom, repeating this process repeatedly.

⑤ *ball and box collide and fall onto the floor*: There is a green ball in mid-air, and a wooden block directly below it. Both objects are in free fall. The ball and the block will collide once in mid-air. Then the block will hit the ground first, and the ball will land on the block and bounce several times.

⑥ *boxes collide and bounce apart*: There are two wooden crates and two stones on the ground. The two stones are at rest. The two wooden crates move towards each other, collide, bounce off, and slide until they come to rest due to friction. They do not collide with the stones.

More dataset statistics are as follows:

- Number of scenes: 6.
- Average number of moving interaction objects: 2.
- Average number of static interaction objects: 2.
- Average video length: 3.33 seconds.
- Total number of cameras per scene: 20.
- Frames Per Second (FPS): 30.
- Number of training views: 17.
- Number of novel views: 3.
- Training time: first 50% of each video (involving object interactions).
- Testing time: remaining 50% of each video (involving object interactions).

A.7 More Details of Real-world Object Interaction Dataset

We capture 6 dynamic scenes with 20 GoPro cameras. We leverage audio signals to synchronize all video sequences. COLMAP is applied to the first frames of the 20 captured views to initialize camera poses. As all cameras remain static with stabilization enabled, their extrinsic parameters stay consistent across all timestamps. Detailed dynamics for each scene are as follows:

- As shown in the first row in Figure 8, a cube slides on the table with a certain initial velocity, gradually slowing down until it stops due to the friction of the table.



Figure 8: Qualitative examples from the Real-world Object Interaction Dataset.

- As shown in the second row in Figure 8, a yellow ball falls from mid-air, strikes the tabletop, and bounces back up into the air.
- As shown in the third row in Figure 8, a yellow ball rolls from the table onto a green inclined plane; upon reaching the top, it rolls down, and after returning to the table surface, it continues rolling forward.
- As shown in the fourth row in Figure 8, a ping-pong ball rolls up the incline from the bottom and collides with a cart sliding down from the top. The cart then pushes the ball down the incline together.
- As shown in the fifth row in Figure 8, there is a yellow ball and a blue ball on the table; both are rolling forward towards each other with a certain initial velocity. As they roll, the two balls collide on the table and bounce off one another. Subsequently, their velocities reverse, and they move away from each other.
- As shown in the last row in Figure 8, there are two inclined planes on a tabletop, each with a ball positioned at its top. The balls roll down from the tops of the inclines onto the table surface. They collide, then rebound and continue rolling across the table in opposite directions. Upon reaching the bases of the inclines, their movement is obstructed, and they eventually come to a stop.

A.8 More Details of the Selected 3D Scenes from PhysInOne Dataset

RollUpSlope_BalloonTether__bg082__QN5gTR_trajectory: A book with initial velocity moved upward along the inclined surface. Gravity acted parallel to the slope, opposing the motion. Friction between the book and the surface dissipated kinetic energy. The book's velocity decreased until

motion ceased. A balloon attached by a rope moved with the book. Buoyant force acted upward on the balloon, countered by the rope's tension. The balloon's position remained fixed relative to the book throughout the ascent and stop.

ObliqueProjectile_VerticalFall__bg096__oPD1xb_trajectory: The white catapult launched a white ball with an initial velocity along an oblique trajectory. Simultaneously, another white ball began free fall from rest. Gravity accelerated both objects downward. The launched ball, following its parabolic path, intersected the falling ball's vertical descent. Mid-air contact occurred between the two white balls as their paths converged under gravitational influence.

MovingHitsStationary_FrictionStop_BalloonFloat__bg154__PnIX27_trajectory: On the ground, there is a brown television, a red apple-shaped object, and a yellow balloon. The balloon rises into the air, while the television moves forward, collides with the red object, and finally comes to a stop. After the collision, the red object rolls forward and eventually comes to a stop as well.

MovingHitsFixed_MovingHitsMoving_BalloonTether__bg168__hbhjeE_trajectory: On the ground, there is a toy car and a toy bus; a yellow balloon is tethered to the top of the toy bus, and a stationary obstacle sits in front of the two toy vehicles. The toy bus moves forward, pushing the toy car ahead of it; the toy car stops upon hitting the stationary obstacle, and the toy bus also comes to a halt after striking the stopped car.

MovingHitsFixed_FrictionStop_BalloonTether__bg178__ZXkQkR_trajectory On the ground, there is a rolling wooden barrel and a stationary tree stump; suspended in mid-air is a blue five-pointed star tethered to two orange balloons. Pulled upward by the balloons, the star rises while the barrel rolls toward the stump; the barrel collides with the ascending star along the way and finally comes to a stop after striking the stationary stump.

MovingHitsFixed_ElasticCouple_BalloonFloat__bg168__ctvgYL_trajectory: A fire hydrant is tethered to a pillar by an elastic cord; the cord retracts, pulling the hydrant against the pillar, where it comes to a rest. Nearby on the ground, a blue balloon slowly rises into the air.

ElasticFall__bg801__FmtbNo_trajectory: A deformable blue cube falls from mid-air toward a sink; upon colliding with the sink, it rebounds into the air while undergoing deformation.

A.9 Details of Computation Cost

We evaluate the computational cost on a single NVIDIA RTX 3090 GPU. Training for one complete scene takes approximately 4 hours. Our inference and rendering speed is on par with FreeGave (Li et al., 2025b). The peak GPU memory consumption reaches around 22 GB during training, while the peak memory usage for inference is about 8 GB.

A.10 Details of Generalization to Unseen Geometries and Initial Velocities

Table 3 presents a quantitative comparison evaluating our method's generalization. The model is trained on a baseline 3D scene where a ball falls onto a *flat ground plane*, and this pretrained model is directly applied to a modified scene where the ground plane is replaced by an *inclined slope*. We can see that the future frame extrapolation of the falling ball achieves very satisfactory scores, meaning that our model can generalize to the new interaction dynamics against the slope. Note that, the drop in segmentation scores is mainly because the pretrained model fails to precisely segment background pixels, which does not affect the dynamics generalization.

Table 3: Quantitative results for generalization to unseen geometry.

	AP↑	PQ↑	F1↑	Pre↑	Rec↑	mIoU↑	mPSNR↑	mSSIM↑	mLPIPS↓	PSNR↑	SSIM↑	LPIPS↓
flat ground plane (seen)	66.060	56.760	80.130	79.370	81.100	68.110	17.727	0.678	0.471	28.143	0.878	0.191
inclined slope (unseen)	49.650	24.070	34.500	30.720	44.640	27.720	21.142	0.752	0.548	31.346	0.872	0.182

A.11 Sim-to-Real Generalization

We further evaluate whether physics priors learned in synthetic scenes can directly apply to real-world data without fine-tuning. We train the object interaction module exclusively on the synthetic dataset and test it directly on our Real-world Object Interaction Dataset. During adaptation, we freeze the

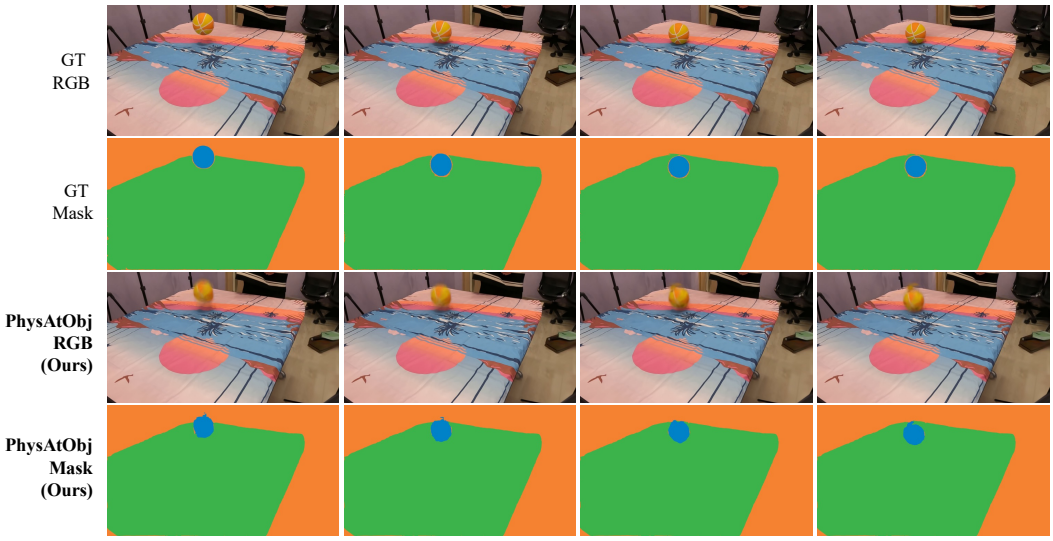

Figure 9: Qualitative examples for sim-to-real generalization.

interaction network and optimize only the scene-specific canonical Gaussians and local deformation fields on unobserved initial frames. As shown in Figure 9, the interaction module can predict real-world object trajectories and collisions, confirming that our model truly captures domain-invariant physical dynamics.

A.12 Analysis of Learned Velocity Changes

Our object interaction module is designed to model impulse-induced velocity changes. To validate this, we visualize the predicted velocity changes during future frame extrapolation in Figure 10. While the ball falls, our framework predicts downward, gravity-driven velocity changes for each Gaussian (green arrows). Upon ground contact, it accurately captures upward velocity changes induced by collision at the contact region (blue arrows). This clearly confirms that our method learns physically meaningful interaction dynamics that generalize well beyond the training window.

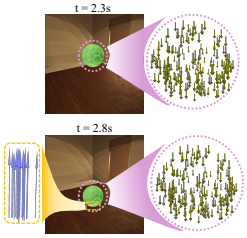

Figure 10: Learned velocity changes.

A.13 Ablation of f_{prop}

Table 4 and Figure 11 compare the results of future frame prediction on a selected 3D scene from PhysInOne, which involves an elastic cube falling and then bouncing upwards after interacting with the sink. We can see that without f_{prop}, the future dynamics of the elastic cube can hardly be predicted.

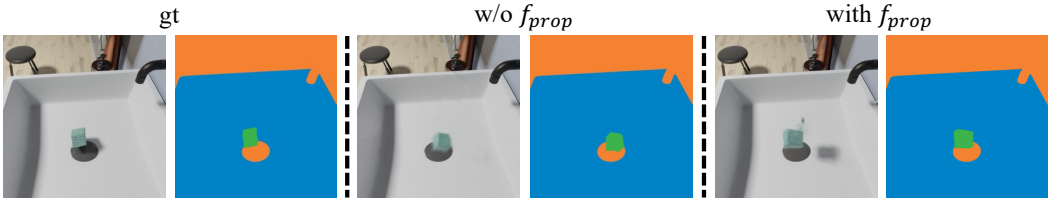

Figure 11: Qualitative results of an ablation study of f_{prop}.

Table 4: Quantitative results of an ablation study of f_{prop}.

		AP↑	PQ↑	F1↑	Pre↑	Rec↑	mIoU↑	mPSNR↑	mSSIM↑	mLPIPS↓	PSNR↑	SSIM↑	LPIPS↓
(8)	w/o f_{prop}	16.551	11.816	22.585	26.013	21.428	17.451	14.844	0.676	0.692	28.732	0.975	0.063
	with f_{prop}	32.009	24.284	49.221	49.935	49.011	36.784	16.854	0.674	0.614	28.878	0.973	0.064

A.14 More Qualitative Results on All Datasets

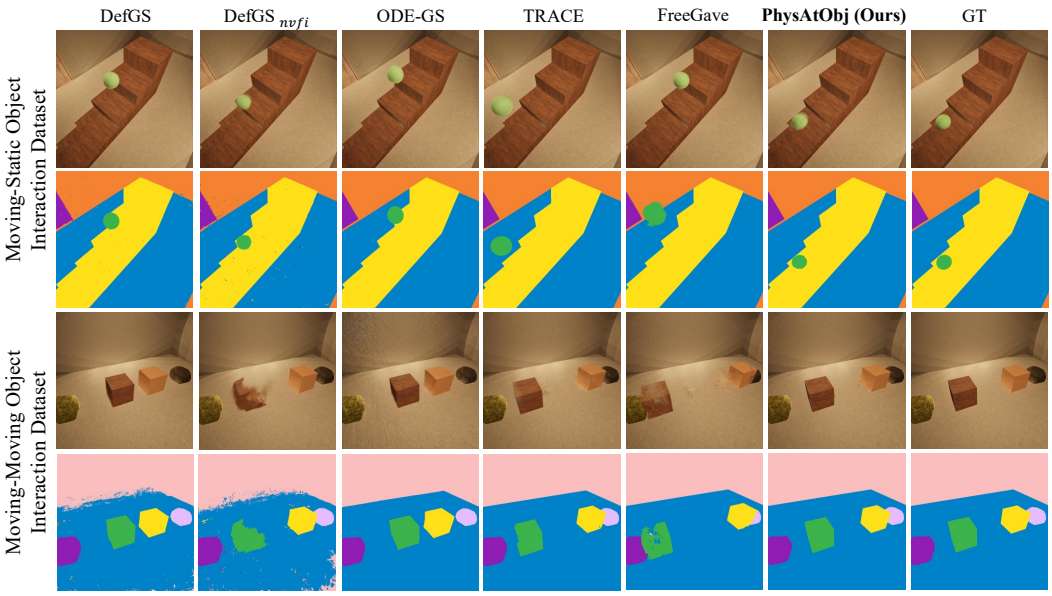

Figure 12: Additional qualitative results.

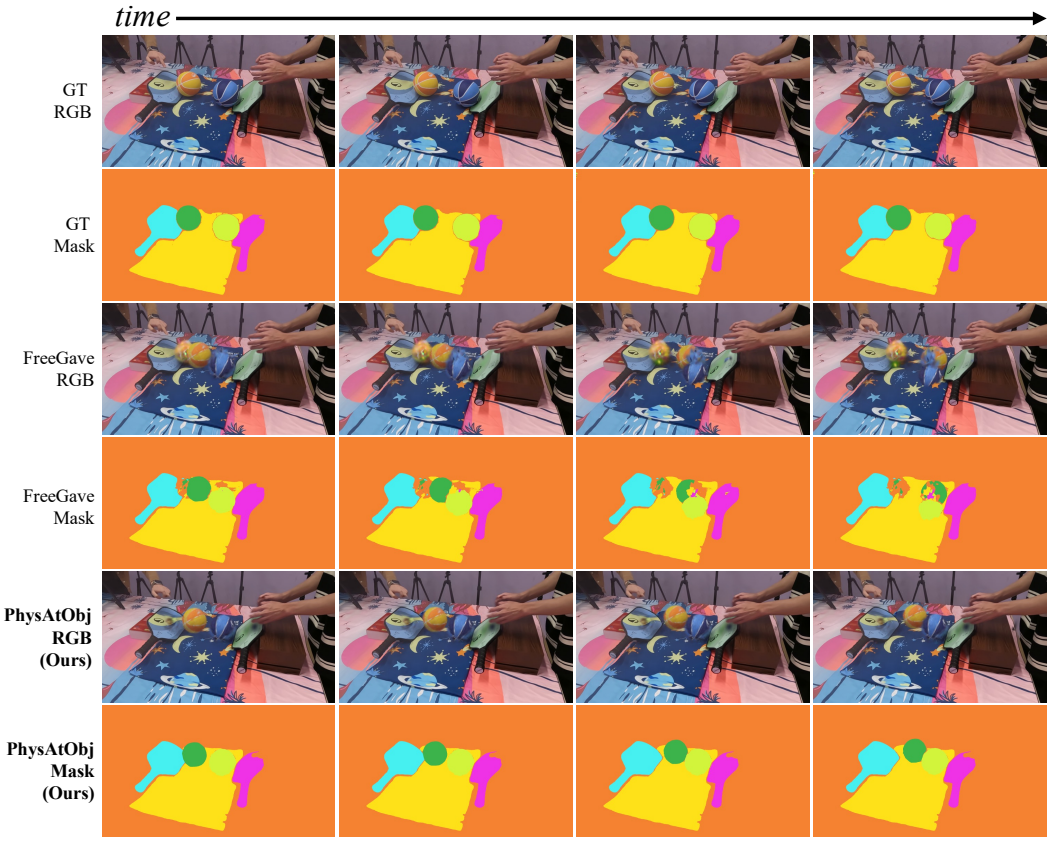

Figure 13: Additional qualitative results.

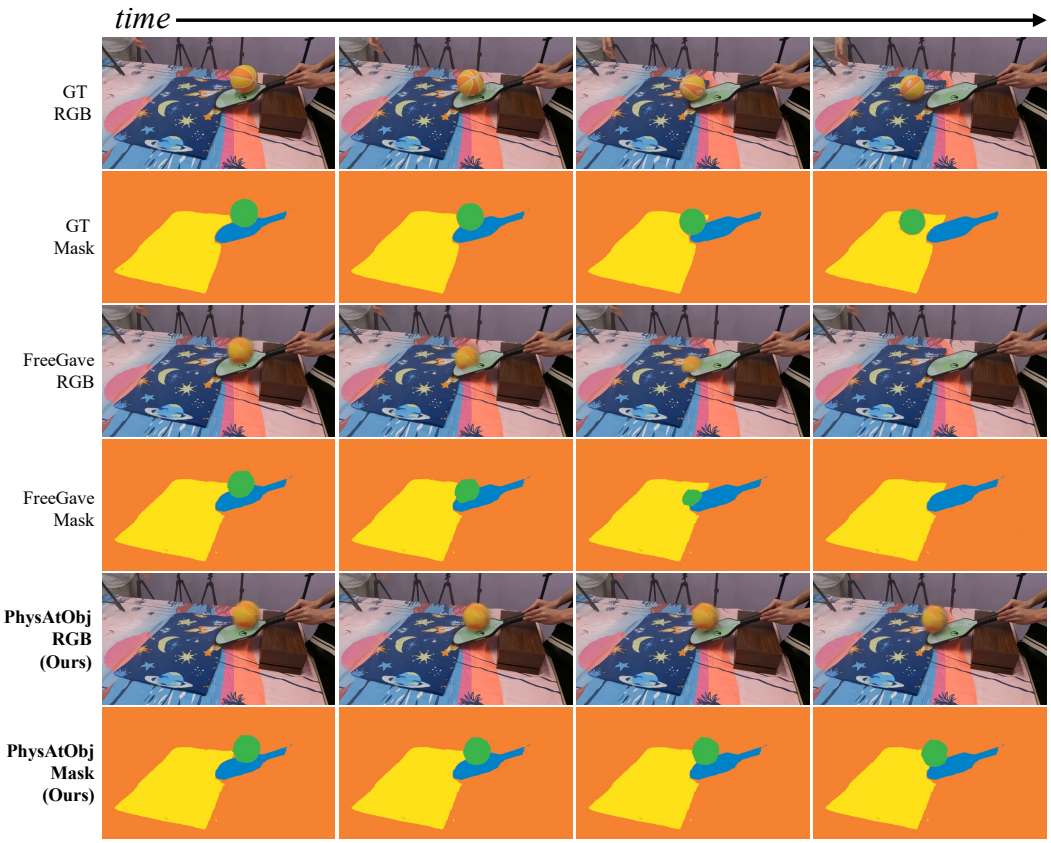

Figure 14: Additional qualitative results.