BEYOND THE VISIBLE: LEARNING DENSE 4D MOTION IN CONTACT-RICH DEFORMABLE WORLDS

Anonymous authors
Paper under double-blind review

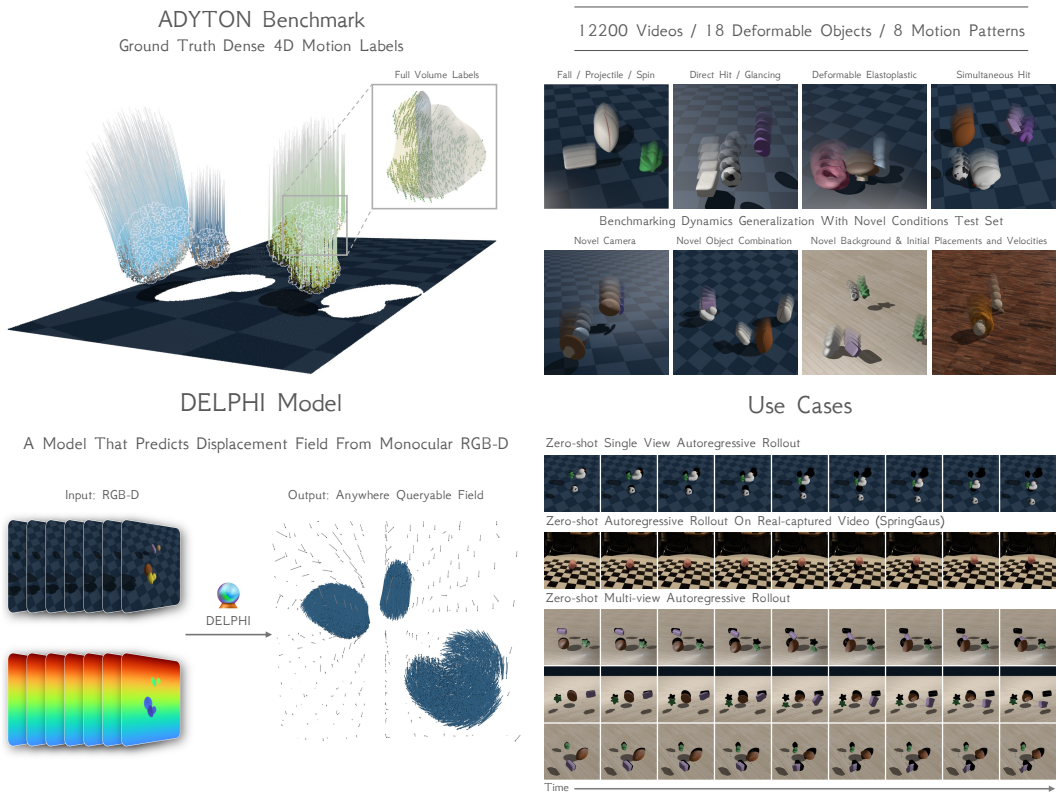


Figure 1: Overview of our contributions: ADYTON serves as a novel 4D dense motion benchmark; DELPHI as a novel method to learn 4D dense motion through monocular RGB-D videos.

ABSTRACT

Predicting 4D dynamics of deformable objects from vision is limited by datasets lacking dense volumetric motion labels and by point-based methods that cannot predict beyond sensor-observed surfaces. We introduce Arena of Deformable dYnamics in Temporal Object Nexus (ADYTON), a large-scale physics-simulated benchmark with ground-truth material-point trajectories throughout the full 3D volume across ten categories of contact-rich deformable interactions, with systematic generalization tests. We propose DEformable dynamics Learned via vision-conditioned Predictive High-fidelity Implicit fields (DELPHI), a vision-conditioned implicit displacement field that decouples spatial queries from input geometry, enabling amodal prediction from monocular RGB-D. On ADYTON, DELPHI matches the strongest baselines on visible surfaces while extending to unobserved regions with negligible degradation, whereas baselines collapse once denied the privileged correspondence that real sensors cannot provide. These results suggest continuous field representations are a more principled foundation for learned 4D dynamics.

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.

1 INTRODUCTION

Dynamics modeling, the capability of predicting future outcomes, is central to general intelligence. It enables machines to plan optimally in response to novel circumstances and learn new skills efficiently in the physical world (Ha & Schmidhuber, 2018; LeCun, 2022). This paper focuses on modeling 4D motions of entities[1] in the scene (Li et al., 2019). Such representations offer direct physical interpretation, thereby effectively support downstream physical tasks like robotics (Liu et al., 2026b). While physics simulation can derive precise 4D motions (Li et al., 2026), applying such approaches at scale is costly: simulation requires identifying material properties in advance (Jiang et al., 2016), but the process involves laborious sensor calibration and human supervision (Si et al., 2024).

Learning-based 4D dynamics models (L4DMs) have emerged as promising alternatives. L4DMs *implicitly* solve the identification problem from past observations (Kovachki et al., 2023; Lu et al., 2021), avoiding manual calibration (Raissi et al., 2019). L4DMs also admit general input modality such as videos or point clouds (Huang et al., 2026), allowing themselves to scale with the availability of visual datasets. However, existing datasets lack either diverse motion data or high-quality motion labels, both are needed for supervising L4DMs. Some datasets leverage simulation to produce ground-truth labels, but primarily consider rigid-body motions and overlook deformable dynamics (Greff et al., 2022a; Zheng et al., 2023a). Others increase dynamics diversity by capturing videos of interacting deformable objects (Zhong et al., 2024a), but videos lack ground-truth motion labels and autonomous annotation tools like 3D tracking (Karaev et al., 2025) remains fragile under occlusion. Beyond data, the lack of rigorous evaluation protocols presents another challenge for L4DMs. Current protocols evaluate only motions of visible surface points, ignoring those inside objects or on occluded surfaces (Jiang et al., 2025). They also lack scrutiny over dynamics modeling along individual dimensions, such as motion patterns or object combinations. Without proper datasets and rigorous evaluation, recent L4DMs fails to model complex, real-world dynamics (Zhang et al., 2025).

We introduce Arena of Deformable dYnamics in Temporal Object Nexus (ADYTON), a simulation benchmark that provides both a large-scale dataset of non-rigid motion trajectories and a principled evaluation protocol for 4D dynamics modeling. ADYTON uses a state-of-the-art simulator (Genesis, 2024) to generate scenes of richly interacting objects and renders them into high-fidelity RGB-D videos. The dataset covers ten types of object interactions spanning contact-free motion, collision, and elastoplastic deformation, with each scene varying in object shapes, textures, material parameters, and the number of objects, ensuring both visual and dynamics diversity. For evaluation, ADYTON measures 4D point motions throughout the entire 3D volume. This assesses 4D dynamics modeling more faithfully than surface-only evaluations. Finally, ADYTON evaluates the generalizability in dynamics modeling along three axes: initial conditions, object combinations, and background appearance, each with a held-out test set.

While ADYTON enables systematic study of 4D dynamics modeling, key design choices for L4DMs remain underexplored. Common approaches model motions of *surface points* (Whitney et al., 2024a; Huang et al., 2026), extracted from RGB-D videos and represented as back-projected pixel maps. Although this design is conceptually simple and enables direct integration with vision foundation models (Wang et al., 2024; Wu et al., 2024b), it tightly couples each output motion to an input sample. The simulation granularity of these methods is limited since the computational cost scales fast with the number of input points. In contrast, *neural flow fields* (Niemeyer et al., 2019) offer a more efficient framework: motion is defined as a function over 3D space and can be queried at any position, decoupled from the input samples. Field-based representations are also general, closely aligned with the continuum description in physics.

We propose DEformable dynamics Learned via vision-conditioned Predictive High-fidelity Implicit fields (DELPHI), a field-based L4DM that takes a monocular RGB-D video as input and predicts a continuous displacement field. It parameterizes the field with visual tokens extracted from the input video, where the motion of individual queried position is retrieved via cross attention (Vaswani et al., 2017), enhanced by learned spatial biases. DELPHI features four properties: (1) it operates on the most general monocular RGB-D input, (2) its output is queryable at arbitrary resolution independent of the input geometry, (3) its inference is highly parallelizable, as predictions at any position is independent of others, and (4) it naturally extends to amodal prediction beyond visible surfaces.

[1] Entity takes different forms. It may refer to a particle, a piece of matter, a part or an object instance.

We evaluate DELPHI on ADYTON benchmark. While DELPHI matches the strongest baselines on visible-surface motion, it additionally predicts amodal motion, an unique capability only field-based representation can achieve. DELPHI also handles long-horizon dynamics modeling: given an initial video, it autoregressively rolls out 4D motions by alternating between dynamics prediction and video rendering. Finally, DELPHI trained with synthetic data generalizes to real-world dynamics on Spring-Gaus (Zhong et al., 2024b): without fine-tuning, it predicts dynamics more accurately than models trained on real data only.

Our contributions are twofold. First, we introduce ADYTON, a benchmark for 4D dynamics modeling with dense motion labels across contact-rich, deformable scenes. It scrutinizes an L4DM's performance along individual axes of initial conditions, object combinations and background appearance. Second, we propose DELPHI, a field-based L4DM that enables full-volume, autoregressive motion prediction from monocular RGB-D input. This capability transfers directly to real-world dynamics.

2 Problem Formulation in 4D Dynamics Modeling

We formulate dynamics modeling as a task of predicting future 4D point motions from past observations of the dynamic scene. We produce point-wise displacements as ground-truth motion labels with physics simulation, generating scenes with diverse dynamics from a fixed set of objects. After training, we evaluate the model's performance along individual axes of initial conditions, object combinations and background appearance. We leave modeling of unseen objects as future work. In the followings, we describe the generation of ground-truth motions and the formulation of L4DMs.

Point-wise displacements as motion labels. To obtain ground-truth motions, we assume that the scene is composed of continuum objects (Lemaitre & Chaboche, 1994). Their dynamics are governed by an underlying velocity field that induces a well-posed flow map[2] (Jiang et al., 2016). With the velocity field, we derive displacements of points in the scene for supervising L4DMs. Let $\mathbf{v}_k : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}^3$ denote the underlying velocity field associated with scene k. The trajectory $\xi_k : \mathbb{R} \to \mathbb{R}^3$ passing through position $x \in \mathbb{R}^3$ at time t satisfies the initial-value problem, given field $\mathbf{v}_k$ and time interval Δt:

$$\dot{\xi}_k(s) = \mathbf{v}_k\big(\xi_k(s), s\big), \qquad \xi_k(t) = x, \qquad s \in [t, t + \Delta t]. \tag{1}$$

In this paper, we interchangeably refer x to either a position or a point, since the position is occupied by the point. From Eqn. 1, we derive the displacement field Φ_k of all points in the scene by integrating the ordinary differential equation (ODE) of the velocity field:

$$\Phi_k(x) \coloneqq \int_t^{t+\Delta t} \mathbf{v}_i\big(\xi_k(s), s\big) ds = \xi_k(t + \Delta t) - x. \tag{2}$$

We obtain point-wise displacements from Φ_k and supervise L4DMs with such labels.

Learning-based dynamics models. An L4DM learns to approximate future point-wise displacements conditioned on past observations of the dynamics scene. Concretely, it trains a neural network $\mathcal{G}_\theta$ with parameter θ on multiple scenes, that takes as input visual observations $\mathbf{o}_k$ and outputs point-wise displacements of each scene k. Based on whether input observations couple with output motions, we separate L4DMs into two forms. The first form models a subset of point-wise displacements, corresponding to the input observations. It writes as: $\mathcal{G}_\theta(\mathbf{o}_k) = \{\Phi_k(x) \mid x \in X(\mathbf{o}_k)\}$ with function X mapping observations $\mathbf{o}_k$ to the set of corresponding points in scene k. Because this framework is conceptually simple and easy to train, it is the most commonly adopted form of L4DMs (Whitney et al., 2024a; Huang et al., 2026).

The second takes a more general form modeling the displacement field function directly: $\mathcal{G}_\theta(\mathbf{o}_k) \longmapsto \Phi_k$. This brings multiple advantages. The output continuous displacement field offers direct physical interpretation for contact detection and deformation analysis, and also complete spatial reasoning for predicting occupancy and motions under occlusion. Meanwhile, field-based approaches support modeling dynamics at any resolution, even with low-resolution input data. This reduces overall computational overhead. Motivated by these observations, we propose a field-based L4DM—DELPHI, and systematically compare it against other baselines for 4D dynamics modeling.

[2]We assume the velocity field is sufficiently regular, such that the induced dynamical system admits unique, non-intersecting particle trajectories, establishing an one-to-one correspondence of points over time

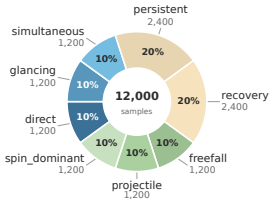
3 Related Work

Learning-Based 4D Dynamics Models. L4DMs predict future object motion from observed scene context and differ in how they represent dynamics. *Scene-flow-based* models (Karhade et al., 2025) and reconstruction methods (Wang et al., 2025) estimate motion and structure over observed frames. *Point-based* models (Whitney et al., 2024b; 2025; Huang et al., 2025; 2026) evolve particle states by scalable point backbones (Wu et al., 2024b; 2025). *Graph-based* models (Battaglia et al., 2016; Li et al., 2019; Sanchez-Gonzalez et al., 2020) learn relational interactions between nodes. *Field-based* models represent motion as spatial functions, such as grid velocities interpolated onto particles (Zhang et al., 2025) or coupled velocity and surface fields (Sang et al., 2025).

Simulation- and Optimization-Based Models. Other methods retain explicit physics or optimize per-scene dynamics. *Physics-integrated* methods (Kairanda et al., 2024; Gaur et al., 2026; Yu et al., 2023; Xie et al., 2024) couple learned representations with thin-shell, fluid, or continuum simulation; *system-identification* methods (Li et al., 2023; Liu et al., 2026a; Jiang et al., 2025) recover geometry and physical parameters from observed motion; and *rendering-based* methods (Lei et al., 2025; Li et al., 2025; Wu et al., 2024a) fit Gaussian motion or velocity fields to video. These priors come at the cost of simulator-specific assumptions or per-scene optimization. Our feed-forward model, trained on ADYTON, predicts spatially queryable dynamics across scenes without test-time optimization.

Table 1: **Left:**ADYTON compared with previous dataset; **Right:** ADYTON motion pattern statistics.

Dataset	Object property	Motion label	Domain
Physion++ (Tung et al., 2023)	Rigid body	surface	sim
Kubric (Greff et al., 2022b)	Rigid body	surface	sim
PointOdyssey (Zheng et al., 2023b)	Articulated objects	surface	sim
Spring-Gaus (Zhong et al., 2024b)	Deformable objects	N/A	real
PhysTwin (Jiang et al., 2025)	Deformable objects	surface	real
ADYTON (ours)	Deformable objects	object volume	sim



4 ADYTON Benchmark Setup

In contrast to physics simulator customizing individual materials, we aim to learn a dynamics model capturing diverse dynamics across materials and scenes. However, training an L4DM requires diverse motion data and high-quality motion labels, both are yet fully provided by existing datasets. Meanwhile, faithful assessment of an L4DM needs to dissect the model's performance along individual dimensions, which is yet supported by current benchmarks. To bridge the gap, we introduce ADYTON, a new simulation benchmark that presents a large-scale dataset and rigorous evaluation protocols for 4D dynamics modeling. Table 1 summarizes the comparison against existing datasets.

ADYTON constructs the dataset in three principles. First, ADYTON enriches interacting dynamics by covering a wide range of object materials and motion patterns. It is built from 18 objects (9 categories) spanning elastic and elastoplastic materials, with motions organized into 10 patterns across three families: contact-free motion, collision, and large elastoplastic deformation. Moreover, object combinations such as co-occurrence patterns, and interactions such as initial positions or velocities, are all randomized per scene. This design not only enhances diversity of dynamics thereby preventing overfitting particular ones, but also supports analysis along the corresponding dimensions.

Second, ADYTON produces high-quality motion labels by simulating scenes with Genesis (2024), a physics simulator built on the material point method (Jiang et al., 2016) for non-rigid dynamics. Within each scene, points are sampled throughout the entire object volume, including visible surfaces, occluded surfaces, and object interiors. These points are tracked with physics simulation, resulting in precise and corresponded motion labels, uniformly covered across the object's 3D volume.

Third, ADYTON renders a monocular RGB-D video with a static camera while introducing strong domain randomization for each scene. Specifically, ADYTON randomizes the camera placement, the scene lighting and the floor texture. This prevents L4DMs from relying on incidental visual clues, but focuses on features relevant to dynamics modeling.

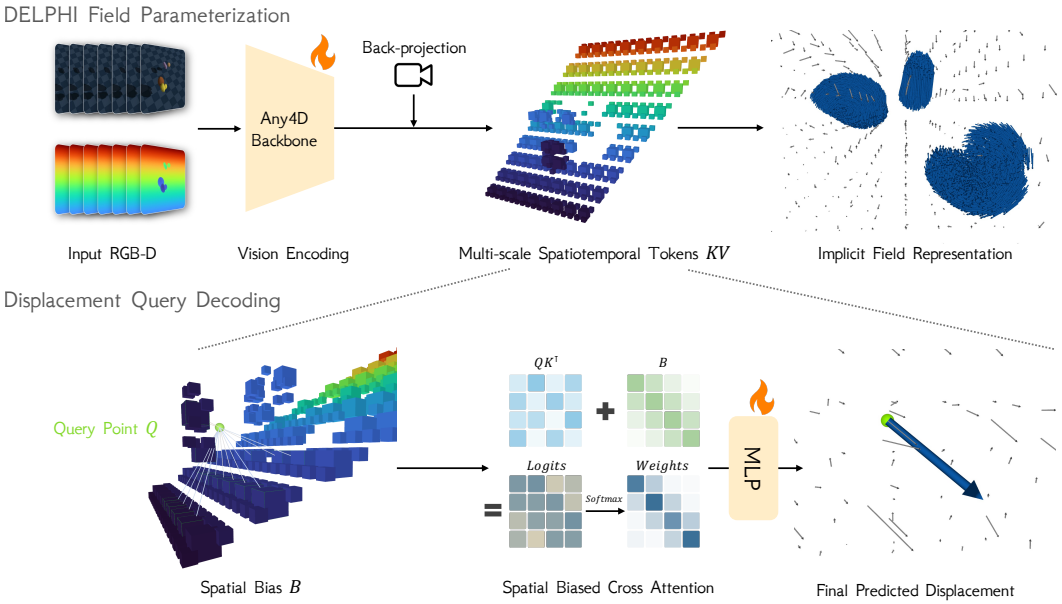


Figure 2: Overview of our proposed DELPHI architecture. DELPHI back-projects the visual tokens and modulate point query with spatial attention bias to predict the displacements.

ADYTON constructs held-out test sets to evaluate an L4DM's performance along three dimensions: novel object combinations (disjoint co-occurrence sets), novel initial-motion conditions, and novel background appearance (photorealistic floors unseen during training). The objects occurring in the test set are sampled from those seen during training. In this paper, we focus on studying whether L4DMs can extrapolate to unseen object compositions. We leave modeling dynamics of entirely unseen novel objects as future work.

In summary, ADYTON comprises 12,200 dynamics scenes (10,000 train / 2,200 test), each consisting of a monocular RGB-D video captured by a static camera together with dense, full volume point-wise displacements, described in camera coordinates. Full details on object materials, motion pattern definitions, and split construction are provided in Appendix B.

5 DELPHI: A VIDEO-TO-FIELD LEARNING-BASED 4D DYANMICS MODELS

We introduce DELPHI, an L4DM for modeling 4D dynamics across diverse materials and scenes. The model takes a monocular RGB-D video as input, and outputs an implicit displacement field that describes motions of all points over the entire 3D volume. This enables high-resolution and spatially complete 4D dynamics modeling from low-resolution inputs, covering point motions on the visible surfaces, occluded surfaces, and even within object interior. Notably, while camera extrinsics are known in ADYTON, DELPHI expresses all motions and locations in the current camera's coordinate system. This avoids camera calibration when applying the framework to other datasets without known camera poses. Fig. 2 shows an overview of the framework.

5.1 MODEL ARCHITECTURE

Inspired by scene representation transformers (Sajjadi et al., 2022; Zhang et al., 2026), we parameterize the output implicit displacement field with a set of visual tokens extracted from the input video, where the motion of individual queried position is retrieved via cross attention (Vaswani et al., 2017).

Representing the implicit field as video visual tokens. We leverage the rich spatial understanding from pre-trained vision foundation models to parameterize the output displacement field. This allows us to accommodate a wide range of vision models with varying designs, without the need for modulating individual architectures. DELPHI uses Any4D (Karhade et al., 2025), that is pretrained on diverse 4D reconstruction tasks thereby encodes rich geometric, motion, and temporal cues relevant to

dynamic scene understanding. Let $\mathbf{o}_k = (I_{t-h}, D_{t-h}, \cdots, I_t, D_t)$ denote a past monocular RGB-D video of a dynamic scene k, where I_s and D_s are the RGB image and depth map at time s, t is the current time, and h is the history length. From the video, we extract visual features per frame, and concatenate them over the horizon, resulting in a set of visual tokens $F_k = (f^1_{t-h}, f^2_{t-h}, \ldots, f^m_t) = \mathrm{Any4D}(I_{t-h}, \ldots, I_t)$, where f^i_s denotes the i-th visual features at time s.

Retrieving point motions via cross-attention. For each point over 3D space, DELPHI retrieves its motions by independently querying the output field with the point's position. This enables input-output decouplings, and makes the inference of a motion field highly parallelizable. Given a query position x, we first encode x with a lightweight MLP into a query token $q = \mathrm{MLP}(x)$, then decode it into a displacement vector by cross-attending to all visual tokens of the input video. The output displacement of the point in scene k is computed as $\mathcal{G}_\theta(\mathbf{o}_k)(x) = \mathrm{MLP}(\mathrm{CrossAttn}(q, F_k))$, where $\mathcal{G}_\theta$ denotes DELPHI with parameter θ.

Relative spatial attention bias. Typical attentions are driven purely by feature similarity between visual tokens and the query token. A query may latch onto salient tokens anywhere in the scene rather than those actually cover the query point. Motivated by the observation that the input RGB-D video already tells us where every token lives in 3D, we use such information to bias each query's attention toward visual tokens that are close to it in space. Concretely, we modulate the attention weight between a query token and a visual token based on their spatial relationships. We obtain each visual token's location using the centroid of back-projected patch, and calculate the offset between the query's position and the visual token's location. The offset is converted into a scalar using a Fourier feature mapping (Tancik et al., 2020) and an MLP, then injected into the attention weight.

Let x be the position of query q and p^i_s be the location of the i-th visual token f^i_s at time s, their attention weight is written as: $\mathrm{softmax}\left(\frac{q^\top f^i_s}{\sqrt{d}} + \phi\big(\gamma(x - p^i_s)\big)\right)$, with a Fourier feature mapping γ and an MLP ϕ. Notably, because the attention considers both the distance and the direction among the query and each visual token, it learns to discover geometric priors well suited to individual dynamics patterns. For example, attending regions below a falling object or in front of a sliding object can better capture the context of these motions than attending to other regions.

Multi-scale visual tokens. Although biasing attentions toward nearby regions help the dynamics model to reason neighboring-object interaction and predict motions within an object more coherently, the model becomes nearsighted and therefore loses the capability of anticipating motions in the far future, that requires reasoning the global context. To address this, we propose to extract fine-to-coarse visual tokens capturing contexts of the dynamics scene at different granularity, and let the query cross-attend to these tokens with spatial biases. We generate coarse visual tokens by aggregating fine tokens with average pooling. Each coarse token's location is the centroid of back-projected pixels within the image patch, unionized from those of the corresponding fine tokens. Empirical results show adopting multi-scale visual tokens can enhance efficiency for dynamics predictions.

We optimize DELPHI to minimize the L2 distance between the predicted and the ground-truth displacements. Since Any4D was pretrained on real-world datasets whose motion patterns and scene layouts differ from those in ADYTON, its visual tokens become less descriptive on our synthetic scenes. To bridge the gap, we apply LoRA (Hu et al., 2022) to the Any4D backbone during training.

6 Experiments

We design experiments to answer: (1) Can DELPHI predict motions over the entire 3D volume from monocular RGB-D input videos? (2) How does DELPHI perform under novel initial motion conditions, object combinations and background appearance? (3) Can DELPHI perform (AR) dynamics prediction? (4) Does DELPHI generalize to real-world videos? (5) Does DELPHI support autoregressive multi-view video rendering? For most experiments, we adopt mean displacement error (MDE) and threshold accuracy Acc_δ with threshold $\delta = 3$ cm as evaluation metrics. The former measures average L2 distance between the predicted and ground-truth per-point displacements, and the latter calculates the percentage of queried points with MDE lower than the threshold.

Experiment 1: Can DELPHI predict motions over the entire 3D volume from monocular RGB-D input videos? We study which of point-based, pixel-based and field-based motion representations is best suited to predicting full volumetric motions from partial observations. We consider 4 point-

Table 2: **Comparison with previous method on visible surface points and full volume prediction.** DELPHI outperforms in full volume while on par with surface points prediction.

	Visible Surface Points		Full Volume	
Method	MDE (cm) ↓	Acc_δ ↑	MDE (cm) ↓	Acc_δ ↑
PGND (Zhang et al., 2025)	1.70±1.26	0.90	5.57±1.14	0.01
PTv3 (Wu et al., 2024b)	1.77±1.25	0.87	4.17±2.18	0.31
Sonata (Wu et al., 2025)	**1.52±0.97**	0.85	4.11±2.16	0.32
PointWorld (Huang et al., 2026)	1.96±1.28	0.81	4.22±2.20	0.30
Any4D (Karhade et al., 2025)	2.83±1.69	0.71	4.39±2.26	0.29
DELPHI (Ours)	1.78±1.01	0.88	**1.92±1.20**	**0.84**

Table 3: **DELPHI generalizes to novel initial conditions.** Each row varying one factor relative to the in-domain baseline.

Condition	MDE (cm) ↓
In-domain	1.92
Novel initial conditions	2.04
Novel object combinations	2.04
Novel background appearance	1.99

Table 4: **DELPHI outperform previous methods in long horizon stability**, @X stands for at step X MDE

Method	@1	@5	@10
PGND	3.58	15.78	30.47
PTv3	2.83	17.15	34.83
Sonata	**1.67**	**10.69**	24.83
PointWorld	2.06	13.57	24.92
DELPHI (ours)	2.23	11.30	**23.86**

based methods: PGND, PTv3, Sonata and PointWorld, which operate on point clouds generated from back-projected RGB-D pixels; we adopt Any4D as the pixel-based approach, taking input as RGB-D videos and outputting 4D motions of observed pixels; we use PGND as a strong field-based baseline producing a queryable motion field from the input point cloud. Since all point- and pixel-based methods have tight input-output couplings, we supervise them with motion of visible surface points. Because they lack the ability to predict amodal motion, we set motion of non-visible points to zero at inference. Both PGND and DELPHI are supervised with motions covered the entire 3D volume.

All results are summarized in Table 2. Our method shines at amodal dynamics prediction: while DELPHI only achieves competitive performance against strong baselines for visible surface points, it substantially outperforms others for prediction motion over the entire 3D volume, gaining 1.92 cm MDE and 0.84 accuracy compared to 4.11 cm MDE and 0.32 accuracy of the second best method. Surprisingly, we also observed that the availability of point correspondences makes huge impacts on point-based methods, and we leave an extensive discussion in the Appendix.

Experiment 2: How does DELPHI perform under novel situations? To assess the model's robustness to various novel scenarios, we create three held-out test sets, each with unseen initial motion conditions (object positions and velocities), unseen object combinations and unseen background appearances during training. As shown in Table 3, DELPHI demonstrates strong robustness, dropping around 4% of the performance given novel initial conditions and object combinations. Most remarkably, changes in background appearance have little impact on our method.

Experiment 3: Can DELPHI perform autoregressive dynamics prediction? To satisfy practical needs for simulation, L4DMs need to predict future dynamics in a long horizon from initial observations. While only trained to predict next-step motion, we can extend DELPHI to an autoregressive dynamics model by alternating dynamics prediction with video rendering. Concretely, we repeat the process of back-projecting image pixels from the last frame of the input video, transporting them in 3D with predicted motions, then rendering a new RGB-D video frame with updated pixel positions. This allows DELPHI to rollout step by step. For evaluation, we randomly sample 120 videos from the test set, and predict motion of image pixels from the initial input step by step. Rolling out predictions 1, 5, and 10 steps forward, we evaluate predicted point motions with MDE. We compare against PGND, PRv3, Sonata and PointWorld, each takes point clouds as input and does not need video rendering. In this experiment, we set the predicted motion of static background points to zero for all

Table 5: **Prior learned from synthetic data provides AR rollout stability on real-captured data.** We render the surfels displaced by DELPHI and report their photometrics for each rollout step.

	PSNR ↑			SSIM ↑			LPIPS ↓		
	@1	@5	@10	@1	@5	@10	@1	@5	@10
Zero-shot	13.41	14.63	12.76	0.463	0.515	0.453	0.356	0.417	0.506
LoRA Fine-tune	13.42	**15.13**	**14.52**	0.461	**0.529**	**0.533**	0.359	**0.409**	**0.460**
Train from Scratch	**13.84**	12.18	12.05	**0.490**	0.361	0.359	**0.342**	0.433	0.480

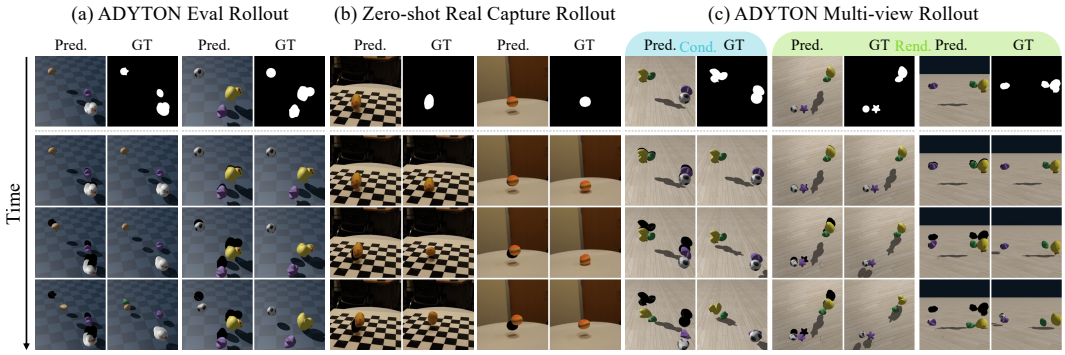


Figure 3: **Qualitative results of AR rollout experiments.** We pick 4 the most difficult scenes to run zero-shot AR rollout. We stack the step at 1,5,10 for each column to show the long horizon results.

methods. Results are shown in Table 4. Our method achieves the lowest MDE for predicting point motions at 10 steps in the future (23.86 vs. 24.83 achieved by the second best baseline).

Experiment 4: Does DELPHI generalize to real-world videos? We study whether simulation data benefits modeling real-world dynamics in two aspects: zero-shot generalization, and dynamics prediction after fine-tuning. Concretely, we compare three setups: (1) testing DELPHI pre-trained in simulation, (2) testing DELPHI pre-trained in simulation and fine-tuned on real-world videos, (3) testing DELPHI trained only on real-world videos. We test on Spring-Gaus dataset (Zhong et al., 2024b), that contains RGB videos of 5 deformable objects falling on the desktop, all unseen from ADYTON simulation dataset. We use an off-the-shelf monocular depth estimator (Yang et al., 2024) to label depths on Spring-Gaus. We adopt a test-time training scheme: training and testing the model on the same set of videos. Because the dataset lacks ground-truth motion labels, we supervise DELPHI with image-rendering loss that optimizes motions of back-projected pixels to reconstruct future video frames, while evaluating models with image-rendering metrics: PSNR, SSIM and LPIPS.

We autoregressively rollout dynamics predictions 1, 5 and 10 steps forward. Results are presented in Table 5. We make two key observations: (1) DELPHI pre-trained on ADYTON directly generalizes to real-world videos, matching the trained-from-scratch counterpart, and (2) fine-tuning simulation-trained DELPHI outperforms training DELPHI from scratch. Altogether, these results demonstrate that scaling simulation data is promising to tackle real-world dynamics prediction.

Experiment 5: Can DELPHI support autoregressive multi-view video rendering? DELPHI outputs a displacement field covering the entire 3D volume, that can be queried with back-projected image pixels from different camera views than the input video, thereby directly supporting multi-view video rendering. We adopt the same autoregressive dynamics prediction setup as Experiment 3, but additionally render multi-view videos. We again clarify that our model takes a single-view video as input, while other camera views only provide RGB colors of points in the 3D space. We present qualitative results inFig. 4, including single-view rendering in ADYTON and Spring-Gaus dataset, and multi-view rendering in ADYTON. We also provide quantitative results in the Appendix that evaluates predicted motions of back-projected pixels from different camera views.

Ablation Study. We analyze the contribution of each model component. We conduct three ablation experiments on ADYTON. Results are summarized in Table 6. First, we study the effect of fine-tuning

Table 6: **Ablations.** MDE (m, ↓) with relative change vs. the baseline column. **(a)** component ablation, single-step and closed-loop rollout. Context length is ablated separately **(b)**, since varying it changes the conditioning window and the resulting rollouts are not step-aligned. **(c)** Token Scales ablation, single-step regime, relative to the default context of 7.

(a) Component ablation

	DELPHI	No LoRA	No spatial bias
@1	2.6	2.2	3.3
@5	1.4	1.5	1.7
@10	2.9	3.2	3.2

(b) Context length ablation

Context	MDE (cm)
7	1.9
4	2.0
1	3.8

(c) Token scale ablation

Scale	MDE (cm)	FPS
1,2,4	1.7	22.20
2,4	1.8	26.45
1	1.6	25.50

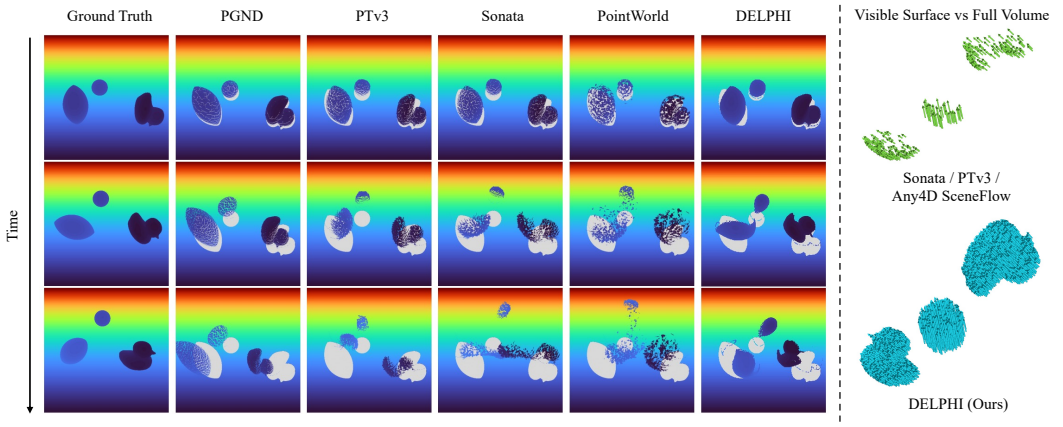


Figure 4: **Qualitative results compared baselines with our DELPHI. Left:** We stack the step at 1,5,10 for each column to show the long horizon results. DELPHI is closest to groud-truth depth L1 map even at the most challenging 10-step rollout horizon; **Right:** Visualized comparison of visible surface point prediction versus full volume prediction.

Any4D backbone with LoRA and modulating attentions with spatial biases. As shown in Table 6a, both components are critical to AR dynamics prediction. Second, we study the impact of number of input video frames, or termed context length. As shown in Table 6b, longer video contexts help the model to capture dynamics more precisely. Third, we verify if fusing multi-scale visual tokens benefits dynamics prediction. As shown in Table 6c, adopting pooled tokens can reduce compute cost albeit negligible performance drop. Here we choose scale 2,4 for balanced FPS and performance.

Visual comparison. We visually compare DELPHI's dynamic predictions with those of baselines. As shown in Fig. 4, each model autoregressively predicts motions of back-projected pixels, and we project these shifted points into monocular depth. DELPHI's prediction is more accurate while capable of retaining the object's original shape. As shown in Fig. 4, DELPHI predicts point motions across the entire 3D volume, whereas point-based methods (Sonata) handles those of input point clouds and pixel-based methods (Any4D) considers only those of observed image pixels.

7 Conclusions & Limitations

We introduced ADYTON, a large-scale simulated benchmark for contact-rich dynamics and DELPHI, a vision-conditioned neural flow field that decouples query from observed points to predict full volume dynamics from single-view RGB-D observation. DELPHI matches point-tracking baselines on visible surfaces and extends to unobserved regions, suggesting continuous fields as a viable alternative to discrete correspondence. **Limitations.** DELPHI predicts a single fixed horizon; autoregressive training with consistency objectives could enable longer stable rollout. Imposing explicit physics regularization such as incompressibility or energy conservation might improve real-world adaptability.

8 Ethics Statement

The authors have read and adhered to the ICLR Code of Ethics throughout the course of this research. This work does not involve human subjects, nor does it use personal or sensitive data that could pose a risk to individuals. All datasets and pretrained models used in this work are publicly available and were used in accordance with their respective licenses. We are not aware of any significant ethical concerns arising from this research.

9 Reproducibility Statement

To facilitate reproducibility, we describe the full synthetic data generation pipeline, including the design of each scenario, in Appendix B. Appendix C details the experimental setup and training procedures for both our method and all baselines, and Appendix D reports the computational resources used in this work. During the review phase, we are happy to provide the codebase to reviewers upon request through an anonymized repository. The codebase and dataset will be publicly released upon acceptance.

10 The Use of Large Language Models

We used large language models (LLMs) as general-purpose writing aids during manuscript preparation, specifically to improve clarity, narrative structure, and LaTeX formatting. LLMs were also used in a limited capacity for coding assistance, including experiment implementation and Slurm job management on our computing cluster; all such code was reviewed and verified by the authors. LLMs were not involved in research ideation, model architecture design, or other core technical contributions, nor in the analysis or reporting of experimental results. All conclusions and insights were formulated and written by the authors, who take full responsibility for the content of this paper.

Bibliography

Peter W Battaglia, Razvan Pascanu, Matthew Lai, Danilo J Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Arihant Gaur, Navami Kairanda, Christian Theobalt, and Vladislav Golyanik. Neuralgarsim: Geometry-agnostic garment simulation with neural fields. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2026.

Genesis. Genesis: A generative and universal physics engine for robotics and beyond, December 2024. URL `https://github.com/Genesis-Embodied-AI/Genesis`.

Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, Thomas Kipf, Abhijit Kundu, Dmitry Lagun, Issam Laradji, Hsueh-Ti (Derek) Liu, Henning Meyer, Yishu Miao, Derek Nowrouzezahrai, Cengiz Oztireli, Etienne Pot, Noha Radwan, Daniel Rebain, Sara Sabour, Mehdi S. M. Sajjadi, Matan Sela, Vincent Sitzmann, Austin Stone, Deqing Sun, Suhani Vora, Ziyu Wang, Tianhao Wu, Kwang Moo Yi, Fangcheng Zhong, and Andrea Tagliasacchi. Kubric: a scalable dataset generator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022a.

Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J. Fleet, et al. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3749–3761, 2022b.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3):440, 2018.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Suning Huang, Qianzhong Chen, Xiaohan Zhang, Jiankai Sun, and Mac Schwager. Particleformer: A 3d point cloud world model for multi-object, multi-material robotic manipulation. In *Conference on Robot Learning (CoRL)*, 2025.

Wenlong Huang et al. Pointworld: Scaling 3d world models for in-the-wild robotic manipulation. *arXiv preprint arXiv:2601.03782*, 2026.

Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. The material point method for simulating continuum materials. *Association for Computing Machinery's Special Interest Group on Computer Graphics and Interactive Techniques 2016 Courses (ACM SIGGRAPH)*, 2016.

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. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Navami Kairanda, Marc Habermann, Christian Theobalt, and Vladislav Golyanik. Neuralclothsim: Neural deformation fields meet the thin shell theory. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Nikita Karaev, Yuri Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker3: Simpler and better point tracking by pseudo-labelling real videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Jay Karhade, Nikhil Keetha, Yuchen Zhang, Tanisha Gupta, Akash Sharma, Sebastian Scherer, and Deva Ramanan. Any4d: Unified feed-forward metric 4d reconstruction. *arXiv preprint arXiv:2512.10935*, 2025.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research (JMLR)*, 24(89):1–97, 2023.

Yann LeCun. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62, 2022.

Jiahui Lei, Yijia Weng, Adam W Harley, Leonidas Guibas, and Kostas Daniilidis. Mosca: Dynamic gaussian fusion from casual videos via 4d motion scaffolds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Jean Lemaitre and Jean-Louis Chaboche. *Mechanics of solid materials*. Cambridge university press, 1994.

Jinxi Li, Ziyang Song, Siyuan Zhou, and Bo Yang. Freegave: 3d physics learning from dynamic videos by gaussian velocity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Minchen Li, Chenfanfu Jiang, Zhaofeng Luo, Wenxin Du, Chang Yu, Žiga Kovačič, and Tianyi Xie. Physics-based simulation, March 2026. URL `https://phys-sim-book.github.io/`.

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. In *International Conference on Learning Representations (ICLR)*, 2023.

Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *International Conference on Learning Representations (ICLR)*, 2019.

Chunjiang Liu, Xiaoyuan Wang, Qingran Lin, Albert Xiao, Haoyu Chen, Shizheng Wen, Hao Zhang, Lu Qi, Ming-Hsuan Yang, Laszlo A Jeni, et al. Mosiv: Multi-object system identification from videos. In *International Conference on Learning Representations (ICLR)*, 2026a.

Zeyi Liu, Shuang Li, Eric Cousineau, Siyuan Feng, Benjamin Burchfiel, and Shuran Song. Geometry-aware 4d video generation for robot manipulation. In *International Conference on Learning Representations (ICLR)*, 2026b.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Mehdi S M Sajjadi, Henning Meyer, Etienne Pot, Urs Bergmann, Klaus Greff, Noha Radwan, Suhani Vora, Mario Lučić, Daniel Duckworth, Alexey Dosovitskiy, Jakob Uszkoreit, Thomas Funkhouser, and Andrea Tagliasacchi. Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning (ICML)*, 2020.

Lu Sang, Zehranaz Canfes, Dongliang Cao, Florian Bernard, and Daniel Cremers. Implicit neural surface deformation with explicit velocity fields. In *International Conference on Learning Representations (ICLR)*, 2025.

Zilin Si, Gu Zhang, Qingwei Ben, Branden Romero, Zhou Xian, Chao Liu, and Chuang Gan. DIFFTACTILE: A physics-based differentiable tactile simulator for contact-rich robotic manipulation. In *International Conference on Learning Representations (ICLR)*, 2024.

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Hsiao-Yu Tung, Mingyu Ding, Zhenfang Chen, Daniel M. Bear, Chuang Gan, Joshua B. Tenenbaum, Daniel L. Yamins, et al. Physion++: Evaluating physical scene understanding that requires online inference of different physical properties. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

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. In *Conference on Robot Learning (CoRL)*, 2024a.

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. In *International Conference on Learning Representations (ICLR)*, 2024b.

William F Whitney, Jake Varley, Deepali Jain, Krzysztof Choromanski, Sumeet Singh, and Vikas Sindhwani. Modeling the real world with high-density visual particle dynamics. In *Conference on Robot Learning (CoRL)*, 2025.

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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024a.

Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler, faster, stronger. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024b.

Xiaoyang Wu, Daniel DeTone, Duncan Frost, Tianwei Shen, Chris Xie, Nan Yang, Jakob Engel, Richard Newcombe, Hengshuang Zhao, and Julian Straub. Sonata: Self-supervised learning of reliable point representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Hong-Xing Yu, Yang Zheng, Yuan Gao, Yitong Deng, Bo Zhu, and Jiajun Wu. Inferring hybrid neural fluid fields from videos. In *NeurIPS*, 2023.

Chuhan Zhang, Guillaume Le Moing, et al. Efficiently reconstructing dynamic scenes one d4rt at a time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Kaifeng Zhang, Baoyu Li, Kris Hauser, and Yunzhu Li. Particle-grid neural dynamics for learning deformable object models from rgb-d videos. In *Robotics: Science and Systems (RSS)*, 2025.

Yang Zheng, Adam W. Harley, Bokui Shen, Gordon Wetzstein, and Leonidas J. Guibas. Pointodyssey: A large-scale synthetic dataset for long-term point tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023a.

Yang Zheng, Adam W. Harley, Bokui Shen, Gordon Wetzstein, and Leonidas J. Guibas. Pointodyssey: A large-scale synthetic dataset for long-term point tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023b.

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

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

APPENDIX

A Extended Discussions on Related Work **14**

A.1 Benchmark Comparison 14

B Benchmark Generation Pipeline **16**

B.1 Dataset Split 16

B.2 Object Pool Configuration 16

B.3 Simulation Pipeline Details 17

B.4 Criticality Scoring 24

C Experiment Configuration Details **25**

C.1 Experiment Setup 25

C.2 Baseline Backbone Variants 26

C.3 Prediction Head 27

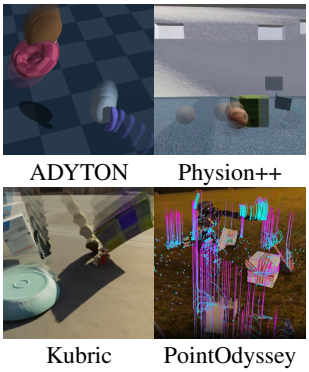
C.4 Optimization Details 27

D Experiments compute resources **27**

A EXTENDED DISCUSSIONS ON RELATED WORK

Table 7: **ADYTON benchmark summary.** The elastic group comprises 5 mesh geometries with 2 stiffness/density variants each (10 objects). The elastoplastic group comprises 4 mesh geometries with 2 yield-stress variants each (8 objects). Within each group, variants share identical geometry and differ only in constitutive parameters. Deformation sub-patterns reuse motion profiles from the Free/Gravity and Contact/Collision families respectively. We refer to Appendix B.2 for more details.

Family	Pattern	Material
Free/Gravity	free fall	Elastic
	projectile	Elastic
	spin dominant	Elastic
Collision	two-object collision	Elastic
	glancing	Elastic
	multi-object collision	Elastic
Deformation	temporal deformation (collision)	ElastoPlastic
	temporal deformation (fall)	ElastoPlastic
	persistent deformation (collision)	ElastoPlastic
	persistent deformation (fall)	ElastoPlastic



A.1 BENCHMARK COMPARISON

Several synthetic benchmarks have been proposed for evaluating physical scene understanding, yet none adequately addresses the challenge of dense motion prediction for deformable materials. Physion++ Tung et al. (2023) primarily focuses on rigid-body interactions with only a limited number of textile examples, and its evaluation protocol is designed around visual question answering rather than dense, per-point motion forecasting. Kubric Greff et al. (2022b) offers massive scale and object diversity, but restricts its scope entirely to rigid-body dynamics, precluding evaluation on soft or deformable materials. PointOdyssey Zheng et al. (2023b) provides large-scale synthetic sequences with long-range point tracking; however, it covers only rigid and articulated objects and lacks clean

per-object motion attribution, making it difficult to disentangle object-level contributions to the observed motion field. In contrast, our benchmark is specifically designed to evaluate dense displacement prediction over a broad range of material behaviors—including elastic, plastic, and granular deformation—with explicit per-point object and material annotations, thereby enabling systematic assessment of models that aim to capture the full spectrum of everyday physical interactions.

B BENCHMARK GENERATION PIPELINE

This appendix documents the complete design of the ADYTON benchmark generation pipeline, covering the dataset split (Sec. B.1), object pool (Sec. B.2), simulation logic (Sec. B.3), and criticality scoring (Sec. B.4).

B.1 DATASET SPLIT

The 12,200 scenes are divided into three splits: Train (10,000), Eval (2,000), and Floor Eval (200). The Eval split holds out novel object combinations by drawing co-occurring objects from buckets disjoint from those seen in training; test scenes may also contain repeated geometry instances, which are excluded from training scenes. The Floor Eval split additionally introduces photorealistic floor textures unseen during training, evaluating OOD generalization along the scene-layout axis. Per-pattern sample counts are listed in Tab. 8.

Table 8: Dataset split statistics. EP = ElastoPlastic. Floor Eval inherits the same per-pattern breakdown as Eval at $\frac{1}{10}$ scale.

Split	Family	Pattern	Samples	Objects	Material	OOD axes
Train	Free/Gravity	free_fall	1,000	3 elastic	Elastic	—
		projectile	1,000	3 elastic	Elastic	—
		spin_dominant	1,000	3 elastic	Elastic	—
	Contact/Collision	direct_hit	1,000	3 elastic	Elastic	—
		glancing	1,000	3 elastic	Elastic	—
		simultaneous	1,000	4–5 elastic	Elastic	—
	Deformation	recovery	2,000	3–4 EP	ElastoPlastic	—
		persistent	2,000	3–4 EP	ElastoPlastic	—
Train total			10,000			
Eval	Free/Gravity	free_fall	200	3 elastic	Elastic	Object combo
		projectile	200	3 elastic	Elastic	Object combo
		spin_dominant	200	3 elastic	Elastic	Object combo
	Contact/Collision	direct_hit	200	3 elastic	Elastic	Object combo
		glancing	200	3 elastic	Elastic	Object combo
		simultaneous	200	4–5 elastic	Elastic	Object combo
	Deformation	recovery	400	3–4 EP	ElastoPlastic	Object combo
		persistent	400	3–4 EP	ElastoPlastic	Object combo
Eval total			2,000			
Floor Eval	Free/Gravity	free_fall	20	3 elastic	Elastic	Object combo + Floor
		projectile	20	3 elastic	Elastic	Object combo + Floor
		spin_dominant	20	3 elastic	Elastic	Object combo + Floor
	Contact/Collision	direct_hit	20	3 elastic	Elastic	Object combo + Floor
		glancing	20	3 elastic	Elastic	Object combo + Floor
		simultaneous	20	4–5 elastic	Elastic	Object combo + Floor
	Deformation	recovery	40	3–4 EP	ElastoPlastic	Object combo + Floor
		persistent	40	3–4 EP	ElastoPlastic	Object combo + Floor
Floor Eval total			200			
Total			**12,200**			

B.2 OBJECT POOL CONFIGURATION

The object pool contains 18 objects organized in three levels: **group** → **geometry** → **variant**. The *elastic* group (5 geometries × 2 variants = 10 objects) is used by Free/Gravity and Contact/Collision families; the *elastoplastic* group (4 geometries × 2 variants = 8 objects) is used by the Deformation family.

Elastic group. Each geometry has an A variant (stiffer, heavier) and a B variant (softer, lighter), both sharing the same mesh but differing in texture. The Young's modulus E and density ρ are intentionally different between variants to represent distinct real-world materials (e.g., a hard leather football vs. a softer recreational ball).

Table 9: Elastic object material parameters.

Label	geometry	E (Pa)	ν	ρ (kg/m^3)
duck_A	duck	5×10^5	0.35	800
duck_B	duck	2×10^5	0.35	600
sphere_A	sphere	8×10^6	0.28	700
sphere_B	sphere	1×10^5	0.32	400
football_A	football	2×10^6	0.35	850
football_B	football	5×10^5	0.38	630
rubber_star_A	rubber_star	2×10^5	0.45	500
rubber_star_B	rubber_star	3×10^4	0.46	400
soap_A	soap	5×10^6	0.30	1100
soap_B	soap	1×10^6	0.38	750

Elastoplastic group. Each geometry has a *recovery* variant (high von Mises yield stress, deforms then returns toward the original shape) and a *persistent* variant (low yield stress, retains the deformed shape). For each geometry, the recovery variant uses a higher von Mises yield stress than the persistent variant, biasing the material response toward elastic recovery rather than permanent plastic deformation. The exact recovery-to-persistent yield-stress ratio is object-dependent and follows the values specified in the object configuration. The Young's modulus E also differs across variants, reflecting distinct physical material presets rather than a single-parameter scaling.

Table 10: Elastoplastic object material parameters.

Label	geometry	E (Pa)	ν	ρ (kg/m^3)	σ_y (Pa)
egg_recovery	egg	2×10^5	0.35	800	80 000
egg_persistent	egg	5×10^5	0.38	800	30 000
mushroom_recovery	mushroom	1×10^5	0.38	1200	50 000
mushroom_persistent	mushroom	3×10^5	0.40	1200	25 000
torus_recovery	torus	5×10^4	0.43	450	25 000
torus_persistent	torus	8×10^4	0.45	450	7 000
worm_recovery	worm	3×10^4	0.45	200	12 000
worm_persistent	worm	6×10^4	0.48	200	4 000

Object selection in the generator. During generation, object sampling is controlled by the execution mode. In `train` mode, the generator first samples unique geometries without replacement and then samples one variant from each selected geometry. Therefore, two objects sharing the same mesh geometry cannot appear in the same training scene. In `eval` mode, objects are sampled with replacement from the eligible pool, so repeated geometries and even repeated variants are allowed. For deformation scenes, an additional `variant_filter` is applied: a `recovery` scene only samples recovery variants, and a `persistent` scene only samples persistent variants. This filter is applied in both train and eval modes, so OOD object-combination evaluation does not mix recovery and persistent materials within one deformation scene.

B.3 Simulation Pipeline Details

Each sample follows a fixed execution sequence: object placement, kick / initial velocity application, camera positioning, and recorded simulation with rendering. All scenes are simulated for 450 recorded steps with timestep $\Delta t = 0.003125$ s and 25 internal solver substeps per step, corresponding to approximately 1.41 s of simulated physical time. The rendered preview is saved at 30 fps, yielding

an approximately 15 s video. The scenario family determines the entire execution path—which functions are called, which parameter ranges are used, and how the camera is placed.

B.3.1 Object Placement

Objects are placed sequentially using a rejection-sampling loop. For each object, up to 500 candidate positions are sampled uniformly within scenario-specific bounds. A candidate is accepted only if its center-to-center distance to every previously placed object satisfies

$$d(i, j) \;\geq\; r_i + r_j + \delta, \tag{3}$$

where r_i, r_j are bounding-sphere radii and $\delta = 0.35$ m is the minimum surface gap. Bounding-sphere radii are estimated from mesh vertices (maximum distance from centroid to any vertex, scaled by the morph scale factor) for mesh objects, or from analytic formulas for primitives. Each object also receives a uniformly random yaw rotation.

Placement bounds. The spatial sampling region is adjusted per scenario to match the physical requirements of each motion type (Tab. 11). Contact/Collision scenes place objects higher to provide sufficient flight distance before collision. Deformation/impact_induced uses a lower range than elastic contact because elastoplastic objects are softer and require less separation. Simultaneous scenes widen the XY bounds to accommodate multiple non-interfering collision pairs.

Table 11: Per-scenario placement bounds.

Scenario	XY bounds (m)	Z range (m)
Free/Gravity (all patterns)	$[-0.8,\ 0.8]^2$	$[0.50,\ 0.90]$
Contact/Collision (elastic)	$[-0.8,\ 0.8]^2$	$[0.90,\ 1.20]$
Deformation / ground_compression	$[-0.8,\ 0.8]^2$	$[0.50,\ 0.90]$
Deformation / impact_induced	$[-0.8,\ 0.8]^2$	$[0.70,\ 1.00]$
Simultaneous (any family)	$[-1.5,\ 1.5]^2$	same as family

Structured placement for simultaneous collisions. The `simultaneous` pattern requires arranging two disjoint collision pairs whose approach corridors do not interfere. This uses a capsule-based placement strategy (Fig. 5):

1. Randomly partition the picked objects into two (source, target) pairs plus any remaining singles.
2. For each pair, sample a random approach axis in the XY plane. If multiple pairs exist, the axis selection is retried (up to 128 attempts) to enforce a minimum inter-pair angle of 35°, ensuring that the two collision corridors are sufficiently separated.
3. Fix each pair's initial center-to-center distance to 0.7 m along its approach axis.
4. Model each pair as a capsule-shaped placement unit—a line segment connecting the two object centers, with radius equal to the larger bounding-sphere radius of the two members.
5. Place capsule units and any remaining single objects using capsule-capsule and capsule-sphere minimum-distance checks. The capsule-capsule distance is computed via the standard closest-point-on-two-segments algorithm.
6. Expand capsule units back into individual object positions.

The same (source, target) pair assignments produced during placement are reused for the kick phase, guaranteeing geometric consistency between initial positions and velocity directions.

B.3.2 Kick / Initial Velocity

Kicks are applied at frame 1 of the simulation. Frame 0 is an initial solver step that allows Genesis to settle internal state (e.g., particle-grid mapping). Two kick functions are used, selected by the scenario family.

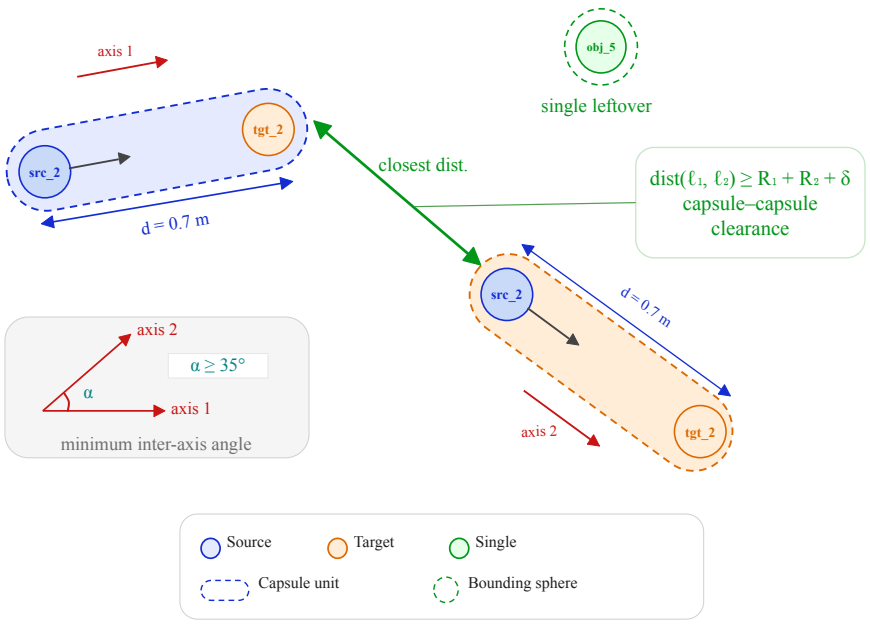


Figure 5: **Capsule-based placement for simultaneous collisions** (top-down schematic, not drawn to scale). Each source–target pair is treated as a capsule whose segment connects the two object centers and whose radius is the larger member radius. The pair distance is fixed to $0.7\,\mathrm{m}$, while different pair axes are constrained to be separated by at least 35° when possible. Capsule–capsule and capsule–sphere clearance tests prevent different collision corridors from overlapping.

Clearance-guided safe kick (Free/Gravity). This function assigns an initial velocity while avoiding object-object contacts, preserving the no-contact semantics of the Free/Gravity family. Despite the function name in code, the selected horizontal direction is not purely "away" from the scene: candidate directions are scored by collision clearance, with a small additional bias toward the scene center to keep objects in view.

1. Sample 96 candidate directions uniformly in the XY plane (with a random angular offset for per-sample variation).
2. For each candidate direction, compute the minimum lateral clearance to every other object's bounding sphere along the travel path up to a horizon distance of $\max(1.6,\ 2.2 \times R_{\text{scene,xy}})$. A direction is rejected if any clearance is negative. A small score bonus of $+0.2$ is added for directions pointing toward the scene center, biasing kicked objects to stay in frame.
3. Select the direction with the highest clearance score among candidates that are both collision-free and not pointing away from the scene center. If no such horizontal direction exists, the implementation falls back to a vertical-only kick by setting the horizontal speed to zero.
4. Compose the velocity from two independent components: horizontal speed sampled from the pattern's speed range and vertical speed sampled from a separate z-speed range (always upward). When spin is enabled, angular velocity $\boldsymbol{\omega}$ is sampled as $\mathcal{N}(\mathbf{0}, \sigma^2_{\text{spin}}\mathbf{I})$, and per-particle velocity is $\mathbf{V} = \mathbf{v}_{\text{lin}} + \boldsymbol{\omega} \times (\mathbf{x} - \bar{\mathbf{x}})$.

Kick toward target (Contact/Collision). This function kicks source objects toward target objects to produce intentional collisions. For each collision pair:

1. *Pair selection with validity scoring.* When pairs are not pre-assigned (non-simultaneous patterns), candidate (source, target, desired_distance) triples are evaluated with a three-level score: 0 = clear path and no post-repositioning overlap, 1 = approach corridor partially blocked by a third object, 2 = overlap with another object after repositioning. The triple with the lowest score is selected. For simultaneous patterns, pairs are pre-assigned during the capsule-based placement phase.

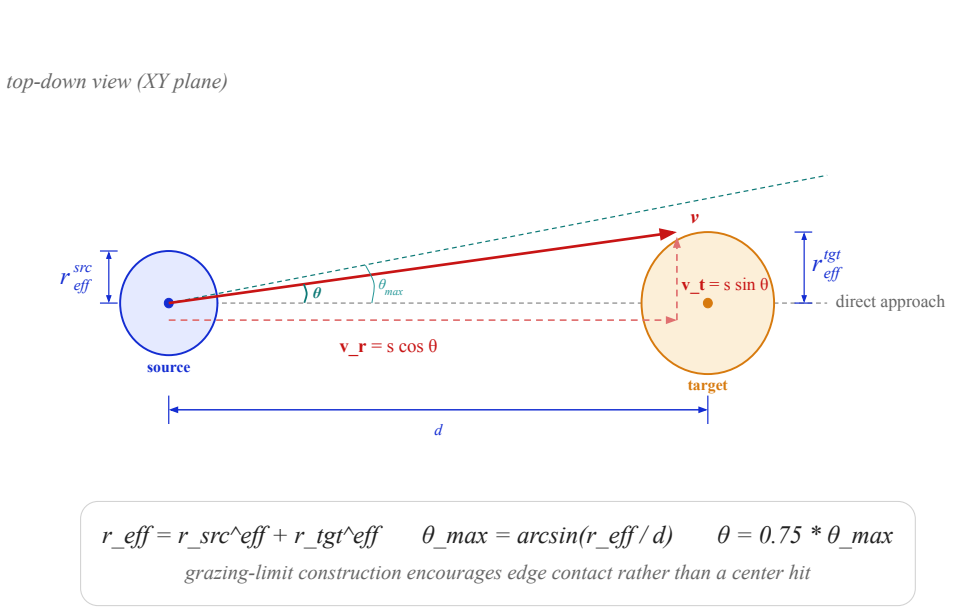


Figure 6: **Glancing deflection geometry** (top-down schematic). The direct source-to-target direction defines the radial axis, and the generator selects one of the two tangential sides based on the larger combined effective extent of the source and target. The sampled speed s is decomposed into radial and tangential components using $\theta = 0.75\,\theta_{\max}$, yielding controlled edge contact rather than a head-on collision.

2. *Repositioning.* If a target distance range is specified, the source object's particles are rigidly translated along the approach line so that the center-to-center distance equals the sampled desired distance before the kick. The position-shift vector is recorded for later replay.
3. *Velocity assignment.* A speed is sampled uniformly from the pattern's speed range and applied in the direction from source center to target center, plus optional spin.

Glancing deflection. For the `glancing` pattern, the velocity direction is deflected away from the target center to produce edge-to-edge contact (Fig. 6):

1. Compute the tangential direction perpendicular to the approach line in the XY plane.
2. Evaluate both tangential sides. For each side, compute the effective radius by projecting source and target particles onto the tangential direction and summing the maximum extents. Select the side with the largest combined effective radius, ensuring robust contact even with non-spherical objects.
3. Let r_{eff} be the combined effective tangential extent of the source and target on the selected side. Compute $\theta_{\max} = \arcsin\big(r_{\text{eff}}/d\big)$, the critical angle at which the deflected trajectory just grazes the combined extent. Set $\theta = 0.75\,\theta_{\max}$ to bias the trajectory toward robust edge contact while avoiding a near-miss trajectory.
4. Decompose the total speed into radial $v_r = s\cos\theta$ (toward target) and tangential $v_t = s\sin\theta$ (side-slip). The final velocity is $\mathbf{v} = v_r\,\hat{\mathbf{r}} + v_t\,\hat{\mathbf{t}}$ where $\hat{\mathbf{r}}$ and $\hat{\mathbf{t}}$ are the radial and chosen tangential unit vectors.

Kick replay for two-pass camera. In contact/collision scenes, the kick is first applied during a non-rendered preview pass (see Sec. B.3.3). The full kick state—linear velocity vector, angular velocity, position shift, and pair assignment—is recorded. During the subsequent recorded pass, these parameters are replayed verbatim via a dedicated replay function with no resampling, guaranteeing that the rendered rollout is identical to the previewed one.

Per-pattern parameter tables. Tables 12 to 14 list the complete parameter ranges for each pattern.

Table 12: Free/Gravity kick parameters.

Pattern	Horiz. speed (m/s)	Vert. speed (m/s)	Spin scale	Num kicks
free_fall	—	—	—	0
projectile	0.5–1.0	0.8–1.5	—	1
spin_dominant	0.3–0.6	0.8–1.2	10.0	1

Table 13: Contact/Collision kick parameters (elastic objects).

Pattern	Speed (m/s)	Glancing	Num kicks	Objects	Target dist (m)
direct_hit	1.5–1.8	No	1	3	0.7–1.0
glancing	1.9–2.2	Yes	1	3	0.7–1.0
simultaneous	1.5–1.8	No	2	4–5	0.7

Table 14: Deformation / impact_induced kick speeds (elastoplastic objects). Persistent variants use higher speeds to compensate for greater energy absorption at low yield stress.

Sub-pattern	Recovery speed (m/s)	Persistent speed (m/s)
direct_hit	1.8–2.1	2.5–2.8
glancing	2.2–2.5	2.9–3.2
simultaneous	1.8–2.1	2.5–2.8

B.3.3 Camera Logic

All scenarios share the same camera placement formula. A random viewing angle is sampled with azimuth $\theta \sim \mathcal{U}(0^\circ, 360^\circ)$ and elevation $\varphi \sim \mathcal{U}(40^\circ, 70^\circ)$. The camera distance is

$$d \;=\; \frac{R}{\tan(\text{fov}/2)} \cdot (1+m) \cdot j, \tag{4}$$

where R is the scene radius, m is the margin, $j \sim \mathcal{U}(\text{jitter range})$, and $\text{fov} = 30^\circ$ for all scenes. All current patterns use $m = -0.08$, meaning the camera is placed *closer* than the minimum fully enclosing distance (Fig. 7). This deliberately allows objects to approach the frame edges, producing tighter shots where objects occupy a larger portion of the image. What differs between families is how R and the lookat point are determined (Fig. 8).

Free/Gravity: analytic trajectory prediction. Since Free/Gravity scenes do not involve planned object-object contact, the generator uses a lightweight analytic proxy to estimate the trajectory extent before recording. This proxy is not a full MPM rollout; it expands the initial particle bounding box using gravity-induced vertical drop and a damped estimate of kick-induced horizontal travel.

1. Dry-run the kick function with `dry_run=True`: this consumes the same random numbers as a real kick (determining direction and speed) but does not modify particle velocities. The RNG state (Python, NumPy, PyTorch) is saved before the dry-run and restored afterward, so the subsequent real kick at frame 1 produces identical results.
2. Compute a conservative bounding box covering the expected trajectory. The vertical drop is estimated from the initial lower bound above the ground as $z_{\text{drop}} = \min(z^0_{\min} - z_{\text{ground}},\ \frac{1}{2}gT^2)$. Horizontal travel is estimated as $\Delta_{\text{xy}} = v_{\text{kick}} T \cdot 0.5$ along the previewed kick direction, where the factor 0.5 is an empirical damping estimate.
3. The lookat point is placed at the bounding-box center, shifted downward by $0.05 \times z_{\text{extent}}$ to bias toward where objects will be after falling.
4. R is set to half the bounding-box diagonal.

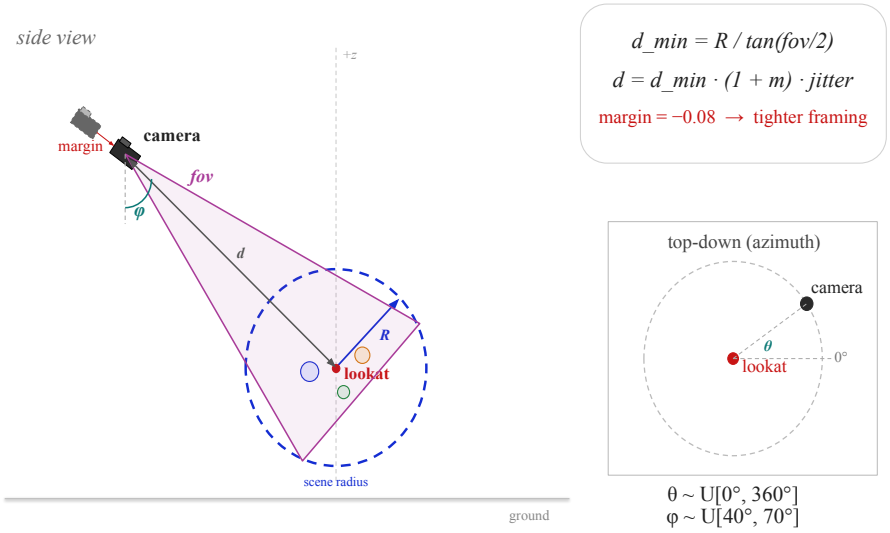


Figure 7: **Generic camera placement geometry**. Given a lookat point and scene radius R, the camera is sampled on a spherical shell with azimuth $\theta \sim \mathcal{U}(0°, 360°)$ and elevation $\varphi \sim \mathcal{U}(40°, 70°)$. The distance follows $d = \frac{R}{\tan(\text{fov}/2)}(1 + m)j$. A negative margin produces intentionally tighter framing.

Contact/Collision: two-pass preview. Post-collision trajectories depend on exact collision geometry and cannot be predicted analytically. The camera must observe the actual trajectory before placing itself.

Pass 1—preview simulation (no rendering):

1. Save the complete RNG state (Python, NumPy, PyTorch).
2. Run one initial solver step (settle), then apply the actual kick and simulate for all remaining steps.
3. At each frame, extract lightweight per-particle position and active mask. Accumulate: (a) the global particle bounding box (element-wise min/max across all frames), (b) a time-weighted centroid (mean active particle position, averaged over frames), and (c) per-object bounding spheres (centroid + max radius from centroid) at every frame.
4. Record the kick parameters for replay.

Camera placement (between passes):

1. Reset the simulation scene and restore the saved RNG state.
2. Re-apply any position shifts that occurred during the preview (scene reset undoes particle positions but the shifts must be preserved for the replay to match).
3. Set the lookat point to the time-weighted centroid, clamped inside the global bounding box. Then recenter: project all per-object bounding spheres onto the camera's view plane (using the sampled θ, φ), compute the center of their projected 2D extent, and shift the lookat in the view plane to center this extent. The shift magnitude is clamped to prevent the lookat from drifting outside the bounding box.
4. Compute R by projecting per-object bounding spheres onto the image plane. For each sphere, effective_perp $= \|\mathbf{p}_\perp\| + r_i$, and $R = \max_i$ effective_perp. This is tighter than projecting the 8 AABB corners because it tracks actual object envelopes rather than the worst-case axis-aligned box.

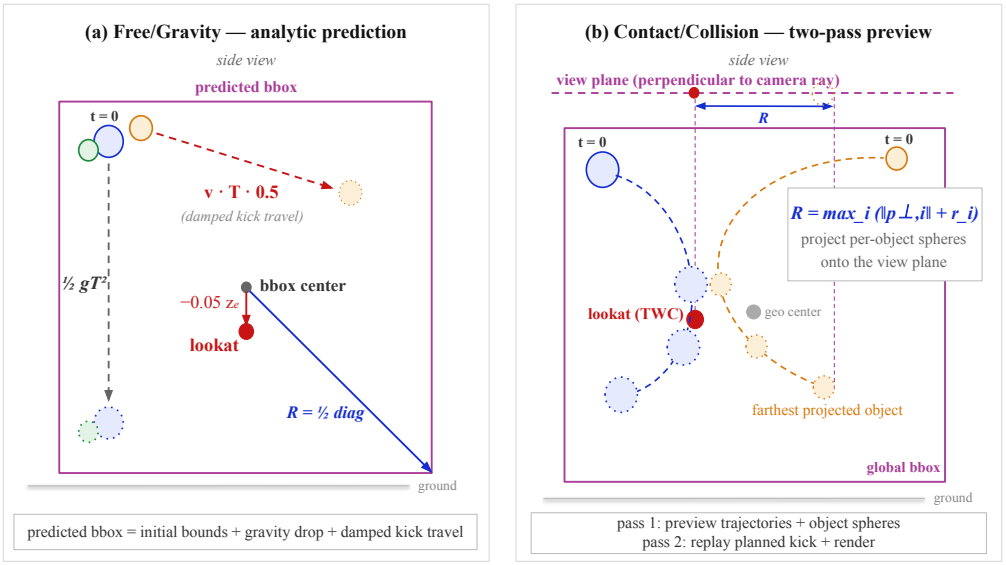


Figure 8: **Lookat point and scene-radius estimation**. **(a)** Free/Gravity scenes use an analytic proxy that expands the initial particle bounding box by gravity drop and damped kick travel; the lookat is the predicted bbox center with a small downward bias, and R is half the bbox diagonal. **(b)** Contact/Collision scenes use a preview rollout to collect the actual trajectory bbox, time-weighted centroid, and per-object bounding spheres. The lookat is initialized from the time-weighted centroid and then recentered in the sampled camera view plane; R is computed from projected per-object bounding spheres rather than from the raw AABB.

Pass 2—recorded simulation (with rendering):

1. Run one initial solver step (same settle as the preview).
2. At frame 1, replay the saved kick parameters—the recorded linear velocity, angular velocity, and position shifts are applied verbatim with no resampling.
3. Run the full simulation with rendering enabled, writing RGB, depth, segmentation, camera parameters, particle positions, object identifiers, optional partial point clouds, and criticality labels to the output HDF5.

Camera profile parameters. Table 15 lists the margin and distance jitter for each pattern. The simultaneous pattern uses a slightly higher jitter lower bound (1.01 vs. 1.00) to provide additional room for scattered collision debris.

Table 15: Per-pattern camera profile parameters.

Pattern	Margin	Dist jitter
free_fall	−0.08	(1.00, 1.04)
projectile	−0.08	(1.00, 1.04)
spin_dominant	−0.08	(1.00, 1.03)
direct_hit	−0.08	(1.00, 1.03)
glancing	−0.08	(1.00, 1.03)
simultaneous	−0.08	(1.01, 1.04)

B.3.4 PARTIAL POINT CLOUD

At each frame, the generator queries the primary camera's renderer for a world-frame point cloud using `render_pointcloud(world_frame=True)`. The returned point cloud is stored at the image resolution $H \times W = 518 \times 518 = 268{,}324$ points per frame and then flattened. A binary validity mask (`partial_mask`) is computed by intersecting the renderer's point-cloud mask with the segmentation map and keeping only object pixels (`segmentation ID` > 1), thereby excluding the ground plane and background. The resulting `partial_points` are camera-visible surface observations in world coordinates, not the full solver particle cloud. Both `partial_points` (float16) and `partial_mask` (uint8) are stored in the output HDF5. The resolution is $H \times W = 518 \times 518 = 268{,}324$ points per frame, stored as flat arrays. A binary validity mask (`partial_mask`) marks pixels belonging to scene objects (segmentation ID > 1), excluding the ground plane and background. Both `partial_points` (world-frame 3D coordinates, float16) and `partial_mask` (uint8) are stored in the output HDF5, providing a camera-centric surface observation complementary to the full solver particle state.

B.3.5 HDF5 OUTPUT MODALITIES

Each generated sample is saved as `data.hdf5`. Table 16 lists all datasets and metadata attributes. Velocity is collected per frame during simulation but is not written to disk; downstream models requiring velocities must derive them from the stored position sequence.

Table 16: HDF5 output schema. T: number of frames; $H{\times}W$: image resolution ($518{\times}518$); N: number of particles (subsampled solver buffer); $HW = H{\times}W$. Multi-camera variants are stored with a `_cam`k suffix on each dataset name.†

Dataset	Shape	Dtype	Description
Camera (primary)			
`rgb`	$(T, H, W, 3)$	uint8	RGB frames
`depth`	$(T, H, W, 1)$	float32	Depth maps
`seg`	$(T, H, W, 1)$	uint8	Segmentation maps
`extrinsics`	$(T, 4, 4)$	float32	Camera extrinsic matrices
`intrinsics`	$(T, 3, 3)$	float32	Camera intrinsic matrices
Particles			
`pos`	$(T, N, 3)$	float16	Solver particle positions
`object_id`	$(N,)$	int32	Per-particle object index (-1 = inactive/padding)
Partial point cloud (optional)			
`partial_points`	$(T, HW, 3)$	float16	Camera-visible surface points (world frame)
`partial_mask`	(T, HW)	uint8	Validity mask (1 = object pixel, 0 = background)
Criticality labels			
`crit_mask`	$(T,)$	uint8	Binary critical-frame indicator
`crit_halo_weights`	$(T,)$	float32	Soft temporal halo weights $\in [0, 1]$
/meta group attributes			
`T`, `H`, `W`, `Nmax`	—	int	Sequence and array dimensions
`fps`, `num_cams`	—	int	Frame rate; number of cameras
`cam_pos`, `cam_lookat`	—	float[3]	Camera position and lookat point
`cam_fov`, `cam_dist`	—	float	Field of view (deg); camera distance
`scene_radius`, `scene_center`	—	float	Scene framing parameters
`scenario_family`, `scenario_pattern`	—	str	Scenario identifiers
`obj`i`_E`, `obj`i`_nu`, `obj`i`_rho`	—	float	Per-object material parameters
`obj`i`_von_mises_yield_stress`	—	float	Yield stress (elastoplastic objects only)

† Additional cameras are stored with a `_cam`k suffix, e.g. `rgb_cam1`, `extrinsics_cam1`, etc.

B.4 CRITICALITY SCORING

Each sample is automatically labeled with per-frame criticality scores that identify frames where significant physical events occur (collisions, large deformations, volume compression). These labels are stored as `crit_mask` and `crit_halo_weights` in the output HDF5.

Input signals. Five per-frame scalar metrics are extracted from the MPM solver's internal state—the velocity gradient $\mathbf{C}$, deformation gradient $\mathbf{F}$, and plastic Jacobian J_p (Tab. 17).

Table 17: Per-frame criticality metrics.

Metric	Formula	Physical meaning
C_norm	$\overline{\|\mathrm{sym}(\mathbf{C})\|_F}$	Strain rate magnitude
C_comp_rate	$\max\big(0,\ -\overline{\mathrm{tr}(\mathrm{sym}(\mathbf{C}))}\big)$	Volume compression rate
F_strain	$\overline{\|\mathbf{F}-\mathbf{I}\|_F}$	Cumulative deformation from rest
J_dev	$\overline{\|J_p-1\|}$	Volume change from reference
J_comp_strength	$\max\big(0,\ 1-\min(J_p)\big)$	Worst-case compression

Normalization. Each metric is z-normalized using an online EMA mean + EMA MAD (exponential moving average of the mean absolute deviation, $\beta = 0.98$), then smoothed with a 5-frame moving average. This makes the score adaptive to per-sample energy scales rather than relying on fixed thresholds.

Dual-head fusion. The five z-scores are combined into two heads and fused:

$$h_{\text{imp}} = 0.5\, z_1 + 0.7\, z_2 + 0.8\, z_5, \tag{5}$$

$$h_{\text{def}} = 0.7\, z_3 + 0.3\, z_4, \tag{6}$$

$$c = \max\big(0,\ 0.65\, h_{\text{imp}} + 0.35\, h_{\text{def}}\big). \tag{7}$$

The impulse head responds to sudden contact events; the deformation head responds to sustained shape change.

Binarization. `crit_mask` is produced by hysteresis thresholding: a frame becomes critical when the fused score exceeds τ_{hi} (90th percentile of the first 30 warmup frames) and stays critical until it drops below τ_{lo} (75th percentile). This avoids rapid on/off flickering around a single threshold.

Temporal halo. `crit_halo_weights` provides a soft $[0, 1]$ weight around each critical frame via exponential decay:

$$w(t) = \exp\Big(-\frac{|t - t_{\text{crit}}|}{\tau}\Big), \qquad \tau = 3.0, \quad K = 8 \text{ frames half-width.} \tag{8}$$

Overflow protection. MPM simulations can produce numerically unstable particles (e.g., extreme deformation gradients). Three layers of protection prevent these from corrupting the score: (1) *Particle explosion filter*: per-frame, any particle with $\max|F_{ij}| > 10^6$ or $\max|C_{ij}| > 10^6$ is excluded from metric computation. (2) *Frobenius norm clamping*: elements are clamped to $\pm 10^{18}$ before computing $\mathbf{A} \cdot \mathbf{A}$, preventing float32 overflow at $\sim 3.4 \times 10^{38}$. (3) *EMA skip + z-score clamping*: non-finite values (NaN/Inf) are skipped in the EMA update; final z-scores are clamped to $[-50, 50]$.

C Experiment Configuration Details

C.1 Experiment Setup

All models are trained on the same dataset and evaluated on held-out subsets. The input consists of H=7 historical frames with stride 10, and the task is to predict the next frame.

Each model predicts a per-point velocity $\mathbf{v} \in \mathbb{R}^3$, and the next-frame position is reconstructed via Euler integration:

$$\hat{\mathbf{x}}_{t+1} = \mathbf{x}_t + \Delta t \cdot \mathbf{v}_{\text{pred}}. \tag{9}$$

We supervise the predicted positions using a Smooth-ℓ_1 loss with β=0.01:

$$\mathcal{L} = \mathrm{SmoothL1}_\beta\big(\hat{\mathbf{x}}_{t+1},\ \mathbf{x}^{\text{GT}}_{t+1}\big). \tag{10}$$

Unless otherwise specified, all models are trained using AdamW (β_1=0.9, β_2=0.999), `bf16` mixed precision (AMP), gradient clipping (max-norm 1.0), and seed 42. Each epoch samples 2,000 training windows.

FULL POINTS TRAINING. In the full-point setting, every history frame contains the same set of N simulated particles, with one-to-one correspondence preserved across different frames. Within a minibatch, samples may have different N, so points are zero-padded to the per-batch maximum $N_{\max}$ during collation; the resulting validity mask is True for all real particles and False only for inter-sample padding. Supervision is applied to every real point against its ground-truth position at $t{+}1$, which is always available because particles are fully observed throughout the trajectory.

VISIBLE POINTS TRAINING. In the visible-point setting, each history frame independently retains only particles whose visibility flag is True at that frame, simulating the realistic regime where a camera observes a partial point cloud per timestep. Per-frame visible counts N_t vary, so each frame is zero-padded to the per-sample maximum $N^{(\mathrm{s})}_{\max} = \max_t N_t$ before being stacked into $[H, N^{(\mathrm{s})}_{\max}, 3]$. Crucially, there is *no correspondence across frames*: particles i in different history frames refers to different real particles. Supervision is paired with the visible particles of the *last* history frame (t): their ground-truth $t{+}1$ positions are queried from the underlying full simulation trajectory , and the loss mask is True only for those last-frame visible particles.

EVALUATION. We evaluate on held-out subsets that span eight scenario categories, treating each category as an evaluation domain. To remove sampling variance and enable strict cross-method comparison, we use a fixed *deterministic, sequential* window sampler rather than the random sampler used at training time: within each domain, every $(C{+}1)$-frame window of every scene is enumerated in canonical order, and the same global random seed is reused across all methods so that the tuple $(\text{domain, batch_idx, sample_idx})$ maps to the *identical* input window for every model. This makes the reported numbers directly paired at the sample level.

We allocate a total budget of 50,000 evaluation windows across the eight domains *proportionally* to their underlying dataset sizes, batched at size 50. The historical context length $H{=}7$ and frame stride match the training configuration. At inference we use the same Euler-integration prediction head $\hat{\mathbf{x}}_{t+1} = \mathbf{x}_t + \Delta t \cdot \mathbf{v}_{\text{pred}}$ with $\Delta t{=}1$, applied without gradient and under `bf16` AMP.

C.2 BASELINE BACKBONE VARIANTS

PTv3 (Pointcept PTv3m1). We use the Pointcept PTv3m1 base configuration with a 5-stage encoder and 4-stage U-Net decoder. Patch sizes follow the default setting: enc_patch_size = (48,48,48,48,48), dec_patch_size = (48,48,48,48), and dec_channels = (64,64,128,256). The input feature dimension is $H{\cdot}3{=}21$, and the decoder outputs 64-dimensional per-point features.

Sonata (Pointcept PTv3m2-Sonata). We adopt the point_transformer_v3m2_sonata configuration with a larger encoder–decoder architecture following the Sonata pre-training setup. The input and output dimensions match PTv3, but model capacity is increased.

PointWorld. We use the official PointWorld sliding-window pipeline with context horizon 7, prediction horizon 1, and window size 8. We convert our dataset into the required WebDataset format and zero-fill missing modalities. The backbone uses a PTv3-based encoder with patch size 256 and predictor dimension 256, producing 64-dimensional per-point features.

PGND (Particle–Grid Neural Dynamics). We use the official PGND architecture. A PointNet encoder over $6H{=}42$ channels per particle (position and velocity at each of the H frames) produces a 64-dim global feature. We disable the table prior and gripper forcing for our object-centric setting.

Any4D. We use the official Any4D feed-forward multi-view transformer, feeding H history frames as RGB-D input and extracting the allocentric scene-flow head to obtain per-point 3D velocity estimates. The egocentric camera-extrinsic head is disabled since our setting operates on point clouds with known camera parameters.

C.3 Prediction Head

We adopt the scene flow head built on top of the backbone features for our scene flow baseline.

For PTv3 and Sonata:

$$\text{LayerNorm}(64) \rightarrow \text{Linear}(64 \rightarrow 256) \rightarrow \text{GELU} \rightarrow \text{Linear}(256 \rightarrow 3).$$

For PointWorld:

$$\text{MLP}(64 \rightarrow 128 \rightarrow 128) \rightarrow \text{Linear}(128 \rightarrow 3),$$

with the final layer initialized using scaled Kaiming-normal weights ($0.1\times$) and zero bias.

For PGND, a grid-conditioned NeRF decoder replaces the per-point MLP: the global feature conditions every voxel of a 50^3 grid (dx=0.02), and a 2-layer MLP (hidden size 64) with positional-encoded voxel coordinates predicts per-voxel 3D velocities that are scattered back to particles by voxel index.

C.4 Optimization Details

Unless otherwise noted, models are trained with learning rate 10^{-4} and weight decay 10^{-4}.

- **PTv3:** batch size 80, trained for 1200 epochs with early stopping.
- **Sonata:** batch size 40, trained for 1200 epochs.
- **PointWorld:** batch size 8, weight decay 0.01, trained for 1200 epochs.
- **PGND:** batch size 128 (64 for the partial-view variant), material network learning rate 5×10^{-5} with gradient-norm clipping at 0.1 and position loss weight λ_x=100.
- **Any4D:** batch size 20, trained for 900 epochs with early stopping.

D Experiments compute resources

We conduct all experiments on AMD Instinct MI300 * 8. A single-seed training for DELPHI takes approximately 2 days.