Back to the Future: REVERSE-CONSEQUENCE MODELING FOR LONG-HORIZON PARTICLE DYNAMICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Accurate prediction of object trajectories is fundamental to mechanical understanding and robotic control. However, autoregressive models inherently suffer from cumulative errors, leading to severe drift in long-horizon rollouts. Observing that in many physical systems with dissipative or convergence-to-rest behavior, the final states are often more predictable than transient dynamics (*e.g.*, a sliding object resting under friction or a pendulum stabilizing at equilibrium), we propose *ReStep*, a cascaded approach that reverses the conventional forward-prediction paradigm. ReStep first employs a stochastic model to directly predict the final state, then leverages *Anchored Diffusion* to generate goal-conditioned intermediate trajectories. By conditioning on both "boundary states", it effectively constrains dynamics and reduces uncertainty, producing physically plausible and temporally coherent transitions. We further integrate ReStep dynamics as the physical engine within a 3D Gaussian Splatting pipeline, enabling a unified physical-visual framework for intuitive inference from images. Extensive experiments on diverse rigid and non-rigid collision datasets show that ReStep achieves superior long-term accuracy, substantially mitigating error accumulation via reverse-consequence modeling.

1 INTRODUCTION

Predicting the evolution of particle-based object dynamics over time is a fundamental challenge in physical simulation (Battaglia et al., 2013; Li et al., 2018; Pfaff et al., 2021; Sanchez-Gonzalez et al., 2018), and is particularly critical for downstream tasks such as visual rendering, motion planning, and robotic manipulation (Chen et al., 2022; Agrawal et al., 2016; Li et al., 2022; Lin et al., 2022; Ma et al., 2022). Existing approaches typically follow an autoregressive paradigm (Han et al., 2022a), where future states are sequentially predicted based on previously generated frames. While conceptually straightforward, these methods inherently suffer from **error accumulation**, such that small deviations at each timestep compound over time, causing trajectories to gradually veer off course and deviate from physically consistent dynamics. Therefore, they often fail to capture long-range dependencies or convergence behaviors common in real-world physical systems, such as frictional settling or contact stabilization.

We observe a notable property in dissipative physical scenes that exhibit convergence behaviors: *The final state, namely the steady state, is reasonably more predictable for neural networks to infer than the detailed transient trajectory leading to it*, under initial conditions and constraints. For instance, when an object falls or collides, its eventual resting configuration is largely determined by global constraints such as gravity, mass distribution, and surface topology. This insight motivates a fundamental question:

What if we predict the outcome first, and then generate the trajectory backwards?

Inspired by this, we propose ReStep, a two-stage approach that reframes the conventional forward-prediction paradigm as a backward-conditioned generation task. As shown in Figure 1, ReStep first employs a stochastic steady-state predictor based on VAE (Kingma & Welling, 2013) to estimate a plausible outcome directly from the initial conditions. Unlike previous particle dynamics prediction methods, such as SGNN (Han et al., 2022a), which do not account for the inherent uncertainty in physical interactions and can yield disentangled predictions over long horizons, ReStep explicitly models the distribution of plausible final states once the dynamic process has stabilized.

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.

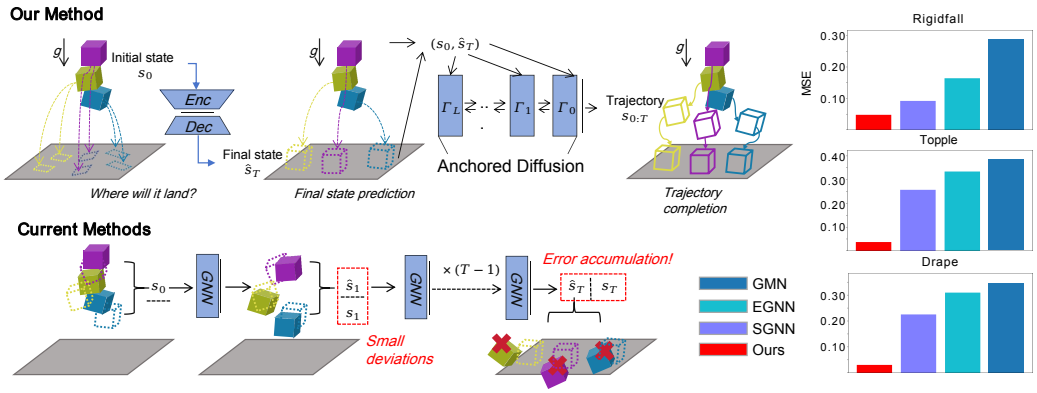


Figure 1: *Overview of the proposed ReStep framework.* **Top:** A VAE-based steady-state predictor estimates plausible outcomes, which drives the Anchored Diffusion model to generate complete trajectories. **Bottom:** Traditional autoregressive models. Our approach mitigates this issue by conditioning trajectory generation on both initial and final constraints. The right-hand bar chart compares the Final State MSE of our method with that of autoregressive methods.

In the second stage, we introduce a *Future-Anchored Diffusion Model* to generate full trajectories conditioned on both the initial and predicted final states, enabling temporally coherent transitions. While autoregressive models tend to accumulate errors over long horizons, diffusion models inherently mitigate this issue through iterative denoising, yielding stable long-term generation. Moreover, their strong generative and representational capabilities facilitate a more comprehensive understanding of complex physical dynamics. The denoising network is designed to be fully convolutional along the particle dimension, enabling flexible processing of variable-sized particle sets. Therefore, Anchored Diffusion can scale seamlessly to a wide range of object and particle counts.

Beyond particle-space trajectory prediction, we note that simulation particles and their physical parameters are often unobservable in real-world settings. To bridge this gap, we integrate ReStep as the physical engine within a 3D Gaussian Splatting (3DGS) pipeline. By coupling future-anchored dynamics with high-fidelity rendering, our approach enables direct inference from images, supporting tasks such as latent property estimation and physically consistent visual dynamics generation.

We conduct experiments on the *FlexiFall* and *Physion* benchmarks, covering both particle-based dynamics prediction and vision-grounded tasks for rigid and non-rigid objects. Our results show that ReStep improves long-horizon prediction accuracy, enables effective recovery of latent physical parameters from visual observations, and produces visually faithful scene dynamics by mitigating cumulative errors through reverse-consequence modeling.

In summary, our key contributions are as follows:

- We propose ReStep and reformulate long-horizon trajectory generation via reverse-consequence modeling, anchoring predictions on inferred future states to effectively eliminate cumulative drift.
- We develop a scalable VAE-based predictor that encodes physical interactions via GNNs and models the multimodal distribution over plausible future steady states.
- We introduce Anchored Diffusion, a future-conditioned generative model that synthesizes temporally coherent particle trajectories and improves long-horizon rollout accuracy.
- We incorporate ReStep as the dynamics module in 3D Gaussian Splatting, enabling intuitive inference of latent physical properties directly from raw images.

2 Related Work

GNN-based dynamics modeling. Graph neural networks (GNNs) (Sperduti & Starita, 1997; Hamilton et al., 2017; Gilmer et al., 2017) have shown to be powerful for modeling particle-based physical systems (Li et al., 2018; Huang et al., 2022; Satorras et al., 2021; Sanchez-Gonzalez et al., 2020; Han et al., 2022a; Xu et al., 2024; Liu et al., 2024). Specifically, the graph is constructed by considering the pairwise distances among particles. Its evolution over time is derived through several iterations of message passing, where each node is updated by aggregating information from its neighbors. Some works (Huang et al., 2022; Satorras et al., 2021; Han et al., 2022a) leverage

equivariant graph neural networks (Han et al., 2022b; Xu et al., 2024; Liu et al., 2024) to encode the inherent symmetries of 3D Euclidean space, achieving state-of-the-art performance in predicting physical dynamics. Despite strong performance in one-step transition prediction, these models suffer from error accumulation during trajectory rollouts.

Diffusion models for conditional generation. Recent advances have introduced various techniques that enable diffusion models to perform conditional generation. Among them, classifier guidance (Dhariwal & Nichol, 2021) is a training-free method that uses a separately trained classifier to steer the sampling process toward the desired condition. Subsequently, classifier-free guidance (Ho & Salimans, 2021) was introduced to remove reliance on an auxiliary classifier by jointly optimizing conditional and unconditional objectives within a single model. It provides flexible control over the strength of conditional guidance during inference, significantly advancing the development of text-to-X generation tasks (Rombach et al., 2022; Ho et al., 2022; Tevet et al., 2023; Liu et al., 2023). The inpainting strategy (Sohl-Dickstein et al., 2015) is applied for goal-conditioned trajectory generation by replacing the sampled values with the conditioning values after each denoising step (Janner et al., 2022; Huang et al., 2023). Conditions can also be injected at all denoising steps for more stable conditioning (Ni et al., 2023; Chi et al., 2025), typically through feature concatenation, Feature-wise Linear Modulation (FiLM) (Perez et al., 2018), or cross-attention (Vaswani et al., 2017) mechanisms.

3 Problem Definition

We consider the problem of generating trajectories given an initial configuration. Let $x_t \in \mathbb{R}^{3N}$ denote the 3D positions of N particles at time t, and let α denote the associated physical attributes such as velocity, friction, mass, and elasticity: $\alpha = \{v, f, m, e\}$. We define the full physical state at time t as $s_t = (x_t, \alpha)$, and the resulting trajectory as $s_{0:T} = \{s_0, s_1, \ldots, s_T\}$.

To model the conditional distribution $p(s_{1:T} \mid s_0)$, Traditional approaches typically adopt an autoregressive formulation, sequentially predicting each subsequent state s_{t+1} conditioned on the current state s_t. Instead, we propose an anchoring mechanism that conditions trajectory completion on the initial state s_0 and a predicted final state $\tilde{s}_T$, namely $p\big(s_{0:T} \mid s_0, \tilde{s}_T\big)$.

Given a dataset of particle-based physical trajectories $\mathcal{D} = \{s_0^{(i)}, \ldots, s_T^{(i)}\}_{i=1}^N$, we aim to: (1) predict a plausible final state $\tilde{s}_T$ from s_0, accounting for uncertainty in attributes and interactions; and (2) complete the trajectory by generating $s_{1:T-1}$ conditioned on $(s_0, \tilde{s}_T)$.

4 Method

The ReStep framework addresses the challenge of long-horizon physical dynamics prediction by reformulating trajectory generation as a future-anchored completion task. This reverse-consequence paradigm ensures that the global structure of the motion is constrained by a predictable future, effectively mitigating long-term drift.

This section details the ReStep architecture and is organized as follows: First, we introduce the *Variational Steady-State Predictor* (Sec. 4.1), where a stochastic GNN-based VAE predicts a physically plausible final state s_T conditioned on the initial state s_0 and physical parameters α. Second, we present *Anchored Diffusion* (Sec. 4.2), which employs a Double-UNet architecture to synthesize a temporally coherent trajectory $s_{0:T}$ by interpolating between the two boundary anchor states. Finally, we discuss ReStep-driven Gaussian rendering (Sec. 4.3), which integrates the predicted dynamics into a 3D Gaussian Splatting pipeline for physical inference and visual generation.

4.1 Variational Steady-State Prediction

To better capture the inherently stochastic nature of physical evolution, we adopt a VAE-style latent prediction architecture that models the multimodal uncertainty of object-level final states (Kingma & Welling, 2013). In many object-interaction scenarios, small perturbations in the initial conditions (such as subtle variations in contact forces, collision angles, or surface friction) may lead to substantially different resting configurations. We encode the initial particle configuration into a latent distribution that explicitly represents the uncertainty over possible futures.

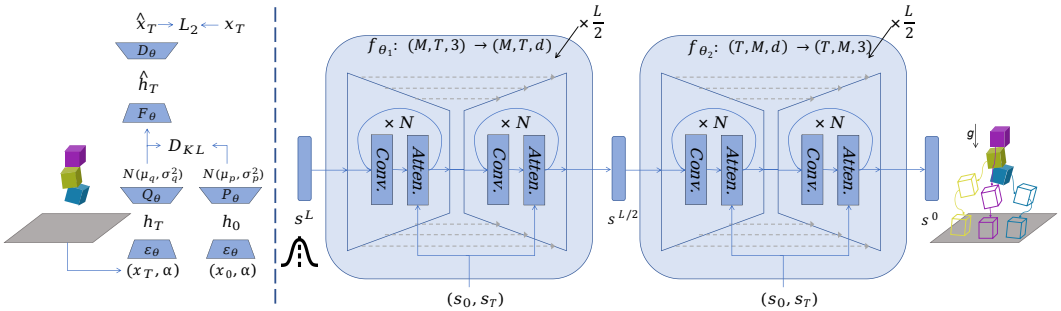


Figure 2: *Model details.* **Left:** VAE-based steady-state predictor. Here, x_T denote the positions at the final state, while x_0 and α represent the positions at the initial state and physical attributes. **Right:** Denoising process. The diffusion-based UNet progressively refines the noisy trajectory s^0, where s_0 and s_T correspond to the initial and final state conditions. M denotes the number of particles and T the length of the trajectory in time. This integration of initial and final state conditions enables trajectory completion.

Our method explicitly models particle-level interactions using Graph Neural Networks (GNNs) (Sperduti & Starita, 1997; Hamilton et al., 2017; Gilmer et al., 2017). Physical systems are inherently relational: the evolution of each particle is strongly influenced by its local neighborhood through contact, collision, and relative motion. GNNs naturally encode such relational inductive biases by performing message passing over dynamically constructed interaction graphs, enabling the model to reason about local geometric structure and inter-particle dependencies in a permutation-invariant manner.

Given an initial physical state $s_0 = (x_0, \alpha)$, represented as a set of particles with positions and physical attributes, we first extract a global latent representation using a GNN-based encoder $\mathcal{E}_\theta(\cdot)$. The encoder constructs a k-nearest-neighbor graph in the particle space and performs multiple rounds of message passing conditioned on relative positions, aggregating local interactions into a global feature vector: $h_0 = \mathcal{E}_\theta(s_0)$. The encoded representation h_0 is then passed to a learned prior network $\mathcal{P}_\theta(\cdot)$, which predicts the parameters of a latent Gaussian distribution: $[\mu_p, \log \sigma_p^2] = \mathcal{P}_\theta(h_0)$.

Meanwhile, during training, a posterior network $\mathcal{Q}_\phi(\cdot)$ is conditioned on the ground truth goal state s_T, which is first encoded using the same GNN-based encoder $h_T = \mathcal{E}_\theta(s_T)$. The posterior network then predicts the distribution parameters of the latent variable: $[\mu_q, \log \sigma_q^2] = \mathcal{Q}_\phi(h_T)$.

We sample $z_T \sim \mathcal{N}(\mu_q, \sigma_q^2)$, fuse it with the initial-state feature through a feedforward latent predictor $\hat{h}_T = \mathcal{F}_\theta([h_0, z_T])$. Finally, a GNN-based decoder $\mathcal{D}_\theta(\cdot)$ maps the predicted latent representation back to the particle space. Conditioned on both the global latent code $\hat{h}_T$ and the initial particle positions, the decoder performs relational message passing and predicts per-particle displacements, yielding the final predicted goal state: $\hat{s}_T = \mathcal{D}_\theta(\hat{h}_T, s_0)$.

The model is trained using a composite objective, denoted by $\mathcal{L}_{\text{goal}}$, which combines a reconstruction loss on the predicted final state with KL divergence regularization that aligns the learned prior and posterior distributions:

$$\|\hat{s}_T - s_T\|_2 + \beta \cdot D_{\text{KL}}\left(\mathcal{N}(\mu_q, \sigma_q^2) \,\|\, \mathcal{N}(\mu_p, \sigma_p^2)\right). \tag{1}$$

This GNN-based latent prediction framework enables the model to capture the multimodal nature of physical outcomes while explicitly accounting for inter-particle interactions and geometric structure. During inference, latent variables are sampled from the learned prior $\mathcal{P}_\theta(\cdot)$ to generate diverse and physically plausible final state hypotheses.

4.2 Anchored Diffusion

We adopt a diffusion-based generative model to synthesize the entire trajectory $s_{0:T}$ given the initial state s_0 and the final state s_T. We will use s to denote the entire trajectory $s_{0:T}$ for simplicity.

Notably, in particle-based physical systems, the number of particles P typically varies with the number of objects or the resolution used to represent them. It is therefore desirable to train a model on varying particle counts, enabling it to capture a broader range of physical dynamics and generalize to

novel scenes, such as those with different numbers of objects. We achieve this with the Double-UNet architecture, which applies convolutions separately along the time and particle dimensions. The first UNet models the temporal evolution of each particle, while the second captures the interactions among particles at each time step.

Formally, the Double-UNet is composed of multiple layers of convolution and attention operations. The convolution layers, denoted by $C(\cdot,\cdot)$, capture temporal evolutions or particle-wise interactions and aggregate information across diffusion steps. The attention layers, denoted by $A(\cdot,\cdot)$, perform cross-attention between the particle states and the conditional initial and final states to inject trajectory constraints, and self-attention among the particle states to capture local interactions and enforce physical consistency. Let $h^{(i)}$ represent the feature at the i-th layer, which can be formulated as:

$$h^{(i)} = C(h^{(i-1)}, l) + A(h^{(i-1)}, s_0, s_T), \quad i = 1 : N, \tag{2}$$

where $h^{(0)} = s^l$ is the noisy input trajectory, the diffusion step l is incorporated by adding its sinusoidal positional embedding to $h^{(i-1)}$, and N is the total number of layers.

The denoising model is trained to minimize the following mean squared error (MSE) between the predicted clean samples and the ground-truth ones at uniformly sampled denoising steps, given by:

$$\mathbb{E}_{l\sim\mathcal{U}\{1,2,\ldots,L\},s\sim\mathcal{D}} \left\| s - f_\theta\left(\hat{s} \middle| s^l, l, s_0, s_T\right) \right\|^2, \tag{3}$$

where $\mathcal{U}\{\cdot\}$ denotes the uniform distribution, $\mathcal{D}$ comprises trajectories with variable sequence lengths and particle counts, s^l is obtained by corrupting the trajectory s with Gaussian noise $\epsilon \sim \mathcal{N}(0, I)$, and f_θ denotes the denoising module that models the conditional distribution $p(s \mid s_0, s_T)$, allowing the generated trajectory to be anchored at both endpoints.

At inference time, we initialize from pure Gaussian noise and iteratively apply the learned reverse transitions to generate the full trajectory. This sampling process yields sequences that connect the initial state s_0 to the final state s_T.

4.3 ReStep-Driven Inverse Rendering

While ReStep operates in particle space, the underlying particles and their physical parameters are often unobservable in real-world scenarios. To bridge this gap, we integrate ReStep as the physical engine within a 3D Gaussian Splatting (3DGS) pipeline. We then extend this framework to two tasks: (1) inferring physical parameters from visual observations, and (2) generating visual dynamics conditioned on initial frames and physical attributes.

4.3.1 Inverse Rendering for Physical Parameter Estimation

Given the initial and final multi-view images of a scene (I_0, I_T), we first reconstruct the 3D scene using a 3D Gaussian Splatting (3DGS) approach (Kerbl et al., 2023). By extracting the centers of the optimized Gaussian primitives, we obtain object-level point clouds that capture the geometry at the beginning and the end of the motion. These reconstructed point clouds are then mapped to the simulator particle representation to produce the initial and final particle states, $\hat{x}_0$ and $\hat{x}_T$. Specifically, we use the learned Gaussian segmentation, compute a bounding box for each object, and then sample simulator particles on and inside the box to approximate its spatial extent. We denote $(\hat{x}_0, \hat{x}_T) \sim R(I_0, I_T)$, where $R(\cdot)$ is the 3DGS renderer. Based on these two states, we aim to recover the physical parameters that best explain the observed motion, including friction, restitution, and mass.

To achieve this, we leverage the pretrained steady-state predictor as a differentiable surrogate for forward physical prediction. Since the module learns a mapping from the initial particle state and physical parameters to the final particle configuration, we can freeze the network and treat the physical parameters α as optimization variables. The inference process is formulated as:

$$\alpha^* = \arg\min_{\alpha\in\Omega} \mathcal{L}_{\text{inv}}\big(G_\theta\big(R(I_0), \alpha\big), R(I_T)\big), \tag{4}$$

where G_θ denotes the pretrained steady-state predictor, Ω is the feasible parameter domain, and $\mathcal{L}_{\text{inv}}$ measures the discrepancy between the predicted and observed final steady states. In practice, $\mathcal{L}_{\text{inv}}$ can be defined as an ℓ_2 loss.

4.3.2 RESTEP-DRIVEN VISUAL DYNAMICS GENERATION

Beyond inverse parameter estimation, our framework also supports visual dynamics generation. Given the initial images of a scene and its physical parameters (can also be estimated using our inverse rendering pipeline), we can simulate new visual dynamic trajectory. Similarly, we first reconstruct 3D scene and map its point clouds to the simulator particles, which are then fed into the ReStep dynamics model to predict future motions. Specifically, we use the steady-state predictor to generate the final state under the given/estimated physical parameters, and use the anchored diffusion dynamics model to synthesize the full trajectory conditioned on the initial and final particle states. The particle trajectory is then converted into motion updates for the Gaussian scene representation. Additional details are provided in the Appendix B.

Applying these updates to the reconstructed 3D scene yields dynamic point-cloud motion and, in turn, the corresponding visual dynamics. Therefore, our method produces temporally coherent visual motion sequences that are consistent with both the observed initial appearance and physical properties. This enables controllable visual dynamics generation, where different physical parameters give rise to different predicted motions and visual outcomes.

5 EXPERIMENTS

5.1 DATASETS

We evaluate our method on two datasets: *FlexiFall* and *Physion*. First, we construct *FlexiFall* by extending *RigidFall* (Li et al., 2020) with additional scenes designed to diversify object interactions and deformation patterns. In addition to the original rigid cube falling setup, we introduce (i) a rigid *Topple* scene featuring sequential contacts, and (ii) a cloth-draping scene in which a non-rigid sheet falls and settles onto rigid cubes. All scenes are simulated using *Kubric* (Greff et al., 2022). Second, we use *Physion* (Bear et al., 2021), which provides a wider spectrum of physical scenarios, including rigid and soft-body collisions, stable multi-object configurations, rolling, sliding, and projectile motion. We summarize the dataset statistics in Table 1.

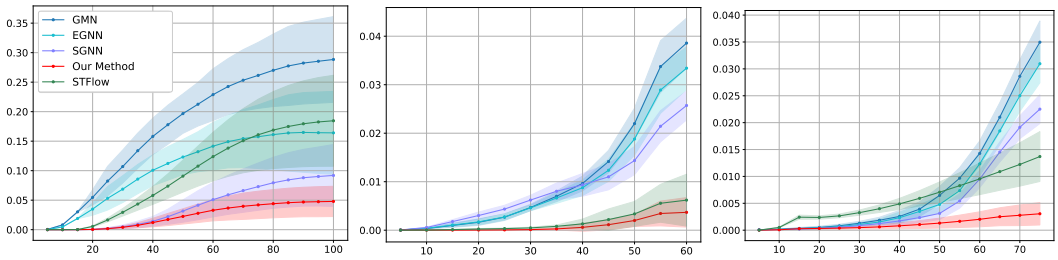
Table 1: *Datasets.* We report dataset volume, horizon length, average number of particles, and average number of objects, respectively.

Scene	#Seq	Horizon	#Par	#Obj
FlexiFall				
RigidFall	10000	150	192	3
Topple	10000	70–150	256	4
Drape	5000	120	800	3
Physion				
Collide	2150	151–452	2383	5
Contain	2150	151–452	8666	7
Drop	2150	151–302	3831	5
Dominoes	2150	151–302	1909	7
Roll	2150	151–302	3819	5

5.2 MAIN RESULTS ON FLEXIFALL

During testing, given the initial state of a scene, the steady-state module performs a single random sampling to predict the final state. The predicted final state, together with the initial state, is then used as the condition for anchored diffusion to generate the full particle trajectory.

We first evaluate our method on the *FlexiFall* dataset, including the original *RigidFall* scene as well as the newly introduced *Topple* and *Drape* scenes. For quantitative evaluation, we compute the mean squared error (MSE) between the predicted and ground-truth particle trajectories, averaged



(a) RigidFall (b) Topple (c) Drape

Figure 3: *MSE comparisons across three FlexiFall scenarios between predicted and ground-truth trajectories.* Our method consistently achieves the lowest error across all time steps.

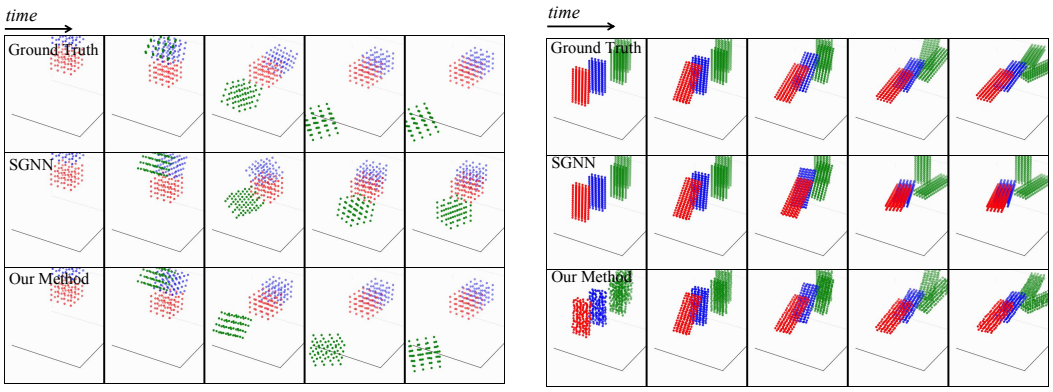


Figure 4: *Prediction showcases on the FlexiFall dataset*. Left: RigidFall scene. Right: Topple scene.

Table 2: *Rollout MSE on Physion for* $t = 50$ *and* $t = 100$. For each scenario, the best-performing method is highlighted in bold, while '–' indicates that the method failed to produce a result at the corresponding horizon.

Method	$t = 50$					$t = 100$				
	Collide	Contain	Drop	Dominoes	Roll	Collide	Contain	Drop	Dominoes	Roll
EGNO (Xu et al., 2024)	1.027	0.111	0.085	0.011	0.444	1.703	0.247	0.107	0.029	0.604
GMN (Huang et al., 2022)	0.981	0.131	0.077	0.012	0.531	1.592	0.331	0.120	–	0.971
SGNN (Han et al., 2022a)	1.009	0.128	0.066	0.012	0.377	1.365	0.310	0.091	0.031	0.616
LatentFlow (Boduljak et al., 2026)	0.105	0.308	0.049	0.005	0.084	0.261	0.454	–	0.014	0.123
Ours	**0.055**	**0.082**	**0.011**	**0.001**	**0.058**	**0.105**	**0.163**	**0.019**	**0.002**	**0.091**

every five time steps. In addition to autoregressive geometric baselines, we implement two recent generative trajectory baselines using the same particle-data interfaces: *LatentFlow*, point-trajectory generation (Boduljak et al., 2026), and *STFlow*, spatio-temporal flow matching with a conditioned random-walk prior (ten Brinke et al., 2026). As shown in Figure 3, our method consistently achieves the lowest MSE across all time steps. Our method preserves long-term stability, resulting in trajectories that are more consistent with the ground-truth temporal evolution. Visualized predicted particle trajectories are shown in Figure 4.

5.3 Main Results on Physion

We evaluate our method on the *Physion* dataset, which provides more complex physical scenarios. We follow the same quantitative evaluation protocol used on the previous dataset. As shown in Table 2, we report the rollout MSE of different methods at two prediction horizons, $t = 50$ and $t = 100$, and additionally evaluate LatentFlow with one generated trajectory per initial condition.

At the long-term horizon ($t = 100$), our method consistently outperforms prior approaches across evaluated scenarios, demonstrating long-term prediction accuracy. This indicates that our model is more effective at mitigating error accumulation over long rollouts. At the shorter horizon ($t = 50$), our method also achieves the best performance across evaluated scenarios, validating its strong short-term predictive capability. These results suggest that ReStep provides accurate and stable dynamics modeling across both short-term and long-term predictions.

5.4 Results on Physics-Grounded Rendering

We evaluate the capability of ReStep for system identification by performing physical parameter estimation on the *Topple* scenario. Leveraging the pretrained steady-state predictor as a differentiable physical surrogate, we invert for three key latent attributes, including friction, restitution, and mass, directly from visual observations. Following the *Kubric* convention (Greff et al., 2022), these attributes are represented as scalar values normalized to the range $[0, 1]$. We construct a visual test set comprising paired scene observations and ground-truth physical parameters. As shown in Table 3,

Table 3: *MSE of physical parameter estimation given visual observations.* We assess friction, restitution, and mass on the *Topple* benchmark.

Method	Fric.	Rest.	Mass
InvGraphics (Jaques et al., 2019)	0.1992	0.1998	0.1719
Ours	**0.0889**	**0.0784**	**0.0796**

Table 4: *Comparison of visual dynamics across three physical scenes.* Metrics are reported for both **in-domain (seen during training)** and **out-domain (unseen future)** time steps. Our model achieves notably lower errors on out-of-distribution time steps, indicating stronger generalization.

Method	Topple			RigidFall			Drape		
	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓
In-Domain									
DeformGS (Yang et al., 2024)	0.9846	43.2263	0.0156	0.9784	35.1682	0.0227	0.9859	41.3140	0.0164
Ours	0.9837	41.9273	0.0171	0.9770	33.3704	0.0215	0.9845	39.0993	0.0157
Out-of-Domain									
DeformGS (Yang et al., 2024)	0.9662	27.4325	0.0551	0.9707	29.0290	0.0335	0.9796	35.2923	0.0226
Ours	0.9763	34.7931	0.0341	0.9824	38.5410	0.0149	0.9837	37.7353	0.0158

ReStep achieves a substantial reduction in estimation error across all parameters, outperforming the Physics-as-Inverse-Graphics baseline (Jaques et al., 2019). Specifically, our method reduces the MSE from 0.1992 to $\mathbf{0.0889}$ for friction, from 0.1998 to $\mathbf{0.0784}$ for restitution, and from 0.1719 to $\mathbf{0.0796}$ for mass. These results demonstrate the effectiveness of our steady-state predictor, which not only improves forward dynamics but also provides more accurate and robust gradients for solving challenging inverse problems.

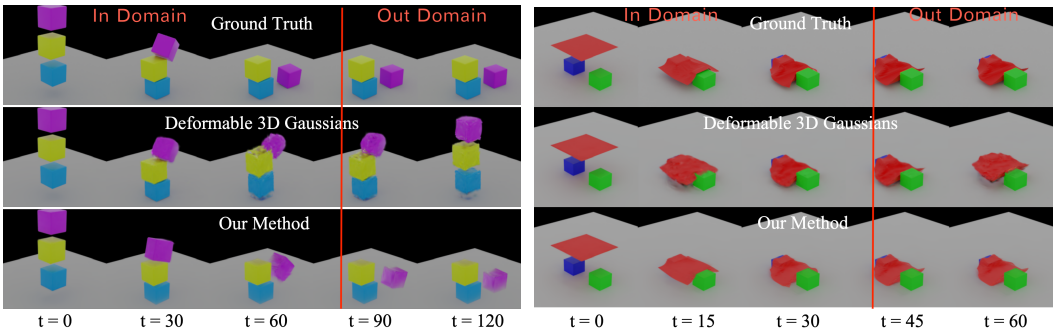
We further evaluate the visual scene generation capability of our framework on the *FlexiFall* benchmark. Starting from a short sequence of initial observations, we predict and render the remaining scene dynamics, and report quantitative results using PSNR, SSIM, and LPIPS. Our method is compared against the Deformable 3DGS baseline, with results summarized in Table 4. Here, "in-domain" refers to time steps within the training distribution, while out-of-domain" denotes long-horizon extrapolation to future time steps beyond the training horizon. Notably, ReStep significantly outperforms Deformable 3DGS in the out-of-domain region. While purely appearance-driven models struggle with non-physical artifacts when extrapolating, our framework maintains high-fidelity structural consistency by leveraging future-anchored physical priors. Qualitative results in Figure 5 further illustrate that ReStep produces physically plausible trajectories that avoid the drift and distortions common in baseline methods.

5.5 Ablation Studies and Model Analyses

5.5.1 Analysis of the Steady-State Predictor

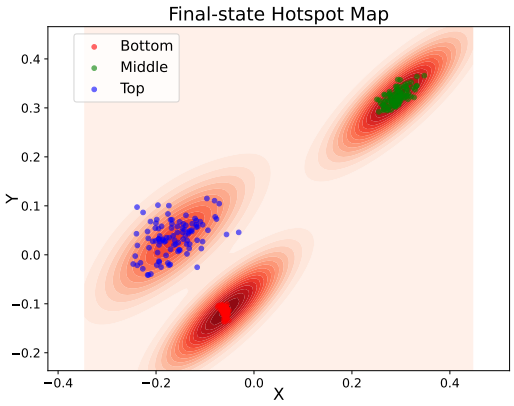
To assess endpoint diversity, we sample 200 trajectories in *RigidFall* and plot each object's final XY-plane centroid (Figure 6a). Objects starting higher exhibit broader endpoint distributions, whereas the lowest object is more concentrated, indicating that the predictor captures differences in outcome uncertainty.

We further assess the reliability of the predicted endpoint distribution using a calibration analysis. The empirical coverage is close to the nominal level at high confidence, while lower-confidence regions exhibit some under-coverage; the full procedure and results are reported in Appendix D.1.

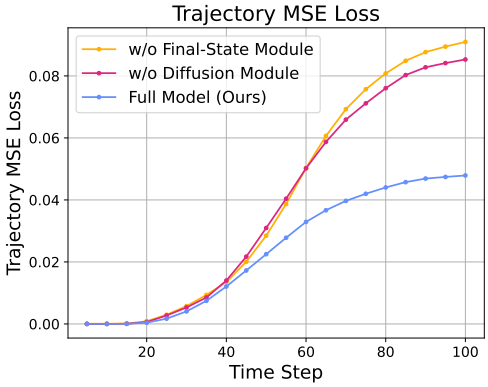


(a) RigidFall (b) Drape

Figure 5: *Comparison of rendered results.* Compared with Deformable 3DGS, our method illustrates that more accurate trajectory predictions for both rigid and non-rigid objects.



(a) *Heatmap of predicted steady-state distributions for three objects in RigidFall.* The highest object exhibits the widest spread and greatest uncertainty, while the lowest is more concentrated, reflecting the expressiveness of our model.



(b) *Ablation studies.* The full model (blue) maintains the lowest error, while removing the steady-state module (yellow) or the diffusion module (red) increases MSE, particularly at later steps. We use SGNN with steady-state inputs as the *no-diffusion* baseline.

Figure 6: *Prediction diversity of the steady-state predictor (a) and ablation results (b).*

5.5.2 Ablation Studies

Figure 6b compares two variants on RigidFall: **(i) w/o steady-state prediction**, a non-autoregressive diffusion model conditioned only on the initial state; and **(ii) w/o anchored diffusion**, an autoregressive SGNN conditioned on the predicted steady state. Both variants increase long-horizon error, particularly the first, supporting the roles of future-state anchoring and diffusion-based synthesis. We further compare VAE-based and deterministic endpoint predictors on Contain and Drop, finding lower mean endpoint MSE with the VAE in both scenes (Appendix D.6). Moreover, Appendix D provides endpoint-condition, backbone, inference-cost, particle-count scaling, and failure-case analyses.

5.5.3 Out-of-Distribution Generalization

We evaluate the out-of-distribution (OOD) generalization under two settings: compositional scene shifts and temporal length variations.

Compositional scene-composition shift. On Topple, we subsample objects from multi-object scenes to create unseen compositions while preserving the interaction rules. ReStep shows strong robustness: the steady-state MSE remains nearly unchanged from 0.0036 ± 0.0020 to 0.0035 ± 0.0019, while the trajectory-level MSE increases only moderately from 0.0010 ± 0.0009 to 0.0019 ± 0.0013. This stability suggests that our GNN-based encoders effectively learn permutation-invariant relational priors that generalize across varying object counts.

Variable trajectory lengths. On Dominoes, we assess generalization to shifts in temporal resolution. We construct OOD samples by linearly interpolating trajectories to approximately twice their original length. Under this significant temporal shift, the average trajectory MSE increases only from 0.0035 ± 0.0020 (in-distribution) to 0.0055 ± 0.0035 (OOD). This modest performance decay demonstrates that the Anchored Diffusion process is not overfitted to specific temporal scales but rather learns a robust, continuous mapping of physical evolution.

6 Conclusions and Limitations

In this work, we presented ReStep, an anchored framework for long-horizon dynamics prediction that conditions trajectory generation on both the initial and inferred final states. By first predicting a physically plausible final state and then generating intermediate states via an anchored diffusion model, ReStep effectively mitigates error accumulation and improves long-horizon stability compared to autoregressive methods. We further extended our framework to visually grounded tasks.

Despite these advantages, ReStep is still sensitive to errors in the predicted steady state, which can propagate through the diffusion model and produce physically implausible trajectories. This could be mitigated by rejection sampling or stronger consistency priors, which could motivate future work on more reliable final-state inference.

AI USE STATEMENT

We used generative AI tools to assist with LaTeX and citation formatting, language editing. The authors reviewed the AI-assisted text. The authors take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

We report the evaluated datasets and scenes, trajectory horizons, and evaluation metrics in Section 5. The model architectures and implementation details are provided in Appendix C, while Appendix D presents additional analyses of endpoint conditioning, model backbones, inference cost, particle-count scaling, and failure cases. We also provide our code in the supplementary materials. For the generative baseline comparison, Physion LatentFlow results use one generated trajectory per initial condition; the corresponding evaluation protocol and unavailable results are identified in Table 2.

REFERENCES

Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. *Advances in neural information processing systems*, 29, 2016.

Peter W Battaglia, Jessica B Hamrick, and Joshua B Tenenbaum. Simulation as an engine of physical scene understanding. *Proceedings of the national academy of sciences*, 110(45):18327–18332, 2013.

Daniel M Bear, Elias Wang, Damian Mrowca, Felix J Binder, Hsiao-Yu Fish Tung, RT Pramod, Cameron Holdaway, Sirui Tao, Kevin Smith, Fan-Yun Sun, et al. Physion: Evaluating physical prediction from vision in humans and machines. *arXiv preprint arXiv:2106.08261*, 2021.

Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *CoRR*, 2023.

Gabrijel Boduljak, Laurynas Karazija, Iro Laina, Christian Rupprecht, and Andrea Vedaldi. What happens next? anticipating future motion by generating point trajectories. In *International Conference on Learning Representations*, 2026. arXiv:2509.21592.

Zhenfang Chen, Kexin Yi, Yunzhu Li, Mingyu Ding, Antonio Torralba, Joshua B Tenenbaum, and Chuang Gan. Comphy: Compositional physical reasoning of objects and events from videos. In *International Conference on Learning Representations*, 2022.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. Pmlr, 2017.

Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3749–3761, 2022.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Jiaqi Han, Wenbing Huang, Hengbo Ma, Jiachen Li, Josh Tenenbaum, and Chuang Gan. Learning physical dynamics with subequivariant graph neural networks. *Advances in Neural Information Processing Systems*, 35:26256–26268, 2022a.

Jiaqi Han, Yu Rong, Tingyang Xu, and Wenbing Huang. Geometrically equivariant graph neural networks: A survey. *arXiv preprint arXiv:2202.07230*, 2022b.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in neural information processing systems*, 35:8633–8646, 2022.

Siyuan Huang, Zan Wang, Puhao Li, Baoxiong Jia, Tengyu Liu, Yixin Zhu, Wei Liang, and Song-Chun Zhu. Diffusion-based generation, optimization, and planning in 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16750–16761, 2023.

Wenbing Huang, Jiaqi Han, Yu Rong, Tingyang Xu, Fuchun Sun, and Junzhou Huang. Equivariant graph mechanics networks with constraints. In *International Conference on Learning Representations*, 2022.

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pp. 9902–9915. PMLR, 2022.

Miguel Jaques, Michael Burke, and Timothy Hospedales. Physics-as-inverse-graphics: Unsupervised physical parameter estimation from video. *arXiv preprint arXiv:1905.11169*, 2019.

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

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Sizhe Li, Zhiao Huang, Tao Du, Hao Su, Joshua B Tenenbaum, and Chuang Gan. Contact points discovery for soft-body manipulations with differentiable physics. In *International Conference on Learning Representations*, 2022.

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*, 2018.

Yunzhu Li, Toru Lin, Kexin Yi, Daniel Bear, Daniel Yamins, Jiajun Wu, Joshua Tenenbaum, and Antonio Torralba. Visual grounding of learned physical models. In *International conference on machine learning*, pp. 5927–5936. PMLR, 2020.

Xingyu Lin, Zhiao Huang, Yunzhu Li, Joshua B Tenenbaum, David Held, and Chuang Gan. Diffskill: Skill abstraction from differentiable physics for deformable object manipulations with tools. In *International Conference on Learning Representations*, 2022.

Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D Plumbley. Audioldm: Text-to-audio generation with latent diffusion models. In *International Conference on Machine Learning*, pp. 21450–21474. PMLR, 2023.

Yang Liu, Jiashun Cheng, Haihong Zhao, Tingyang Xu, Peilin Zhao, Fugee Tsung, Jia Li, and Yu Rong. Segno: Generalizing equivariant graph neural networks with physical inductive biases. In *ICLR*, 2024.

Pingchuan Ma, Tao Du, Joshua B Tenenbaum, Wojciech Matusik, and Chuang Gan. Risp: Rendering-invariant state predictor with differentiable simulation and rendering for cross-domain parameter estimation. In *International Conference on Learning Representations*, 2022.

Fei Ni, Jianye Hao, Yao Mu, Yifu Yuan, Yan Zheng, Bin Wang, and Zhixuan Liang. Metadiffuser: Diffusion model as conditional planner for offline meta-rl. In *International Conference on Machine Learning*, pp. 26087–26105. PMLR, 2023.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International conference on learning representations*, 2021.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, 2022.

Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. In *International conference on machine learning*, pp. 4470–4479. PMLR, 2018.

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*, pp. 8459–8468. PMLR, 2020.

Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International conference on machine learning*, pp. 9323–9332. PMLR, 2021.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015.

Alessandro Sperduti and Antonina Starita. Supervised neural networks for the classification of structures. *IEEE transactions on neural networks*, 8(3):714–735, 1997.

Kiet Bennema ten Brinke, Koen Minartz, and Vlado Menkovski. STFlow: Data-coupled flow matching for geometric trajectory simulation. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, 2026. arXiv:2505.18647.

Guy Tevet, Sigal Raab, Brian Gordon, Yoni Shafir, Daniel Cohen-or, and Amit Haim Bermano. Human motion diffusion model. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=SJ1kSyO2jwu.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Minkai Xu, Jiaqi Han, Aaron Lou, Jean Kossaifi, Arvind Ramanathan, Kamyar Azizzadenesheli, Jure Leskovec, Stefano Ermon, and Anima Anandkumar. Equivariant graph neural operator for modeling 3d dynamics. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 55015–55032, 2024.

Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 20331–20341, 2024.

APPENDIX CONTENTS

A Background . . . 13
B Details of ReStep-Driven Inverse Rendering . . . 14
C Architecture Details . . . 15
D Additional Analyses and Experimental Diagnostics . . . 15

A BACKGROUND

A.1 VAE-BASED PREDICTION

Deep generative models have become a powerful framework for learning complex data distributions. Among them, the Variational Autoencoder (VAE) (Kingma & Welling, 2013) provides a probabilistic formulation that encodes data into a continuous latent space and reconstructs it through a decoder network. Formally, the encoder $q_\phi(z|x)$ maps an observation x to a latent distribution, while the decoder $p_\theta(x|z)$ reconstructs x from the latent variable z. The model is trained by maximizing the

Table 5: Network architectures and hyperparameters for Stage 1: Stochastic Goal State Prediction. All MLP layers use ReLU activations unless otherwise specified.

Module	Input Dim	Output Dim	Architecture Details
Encoder $\mathcal{E}_\theta(\cdot)$	$N \times (3+3)$	z	• Position encoder: MLP($3 \to 128 \to 128$) • Velocity encoder: MLP($3 \to 128 \to 128$) • Feature concatenation to form per-particle embeddings • $L = 3$ graph message-passing blocks with $k = 16$ neighbors • Linear projection: Linear($256 \to z$)
Posterior Network $\mathcal{Q}_\theta(\cdot)$	z	$\mu_q,\ \log \sigma_q^2 \in \mathbb{R}^z$	• MLP($z \to 512 \to 2z$) with normalization • Output split into mean and log-variance
Prior Network $\mathcal{P}_\theta(\cdot)$	z	$\mu_p,\ \log \sigma_p^2 \in \mathbb{R}^z$	• MLP($z \to 512 \to 2z$) with normalization • Output split into mean and log-variance
Frame Predictor $\mathcal{F}_\theta(\cdot)$	$z+z$	z	• MLP($2z \to 512 \to z$) with normalization • Predicts the next latent particle state
Decoder $\mathcal{D}_\theta(\cdot)$	$N \times (z+3)$	$N \times 3$	• Node encoder: MLP($(z+3) \to 256 \to 256$) • $L = 3$ graph message-passing blocks with $k = 16$ neighbors • Output MLP($256 \to 256 \to 3$)

Table 6: Network architectures and hyperparameters for Stage 2: Anchored Diffusion. We use the second UNet as an example, which applies convolution along the particle dimension.

Module	Input Dim	Output Dim	Architecture Details
Convolution Block $C_\theta(\cdot,\cdot)$	(T, M, d_{in})	(T, M, d_{out})	• Conv1d($(T, M, d_{in}) \to (T, M, d_{out})$), SiLU • Feature fusion via element-wise addition • Conv1d($(T, M, d_{out}) \to (T, M, d_{out})$), SiLU
Attention Block $A_\theta(\cdot,\cdot)$	(T, M, d)	(T, M, d)	• SelfAttention($(T, M, d) \to (T, M, d)$) • CrossAttention($(T, M, d) \to (T, M, d)$) • GeGLU($(T, M, d) \to (T, M, 4d) \to (T, M, d)$)

evidence lower bound (ELBO):

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{\text{KL}}(q_\phi(z|x)\|p(z)), \tag{5}$$

which balances reconstruction accuracy and the regularization that enforces the latent space to follow a prior distribution.

Building upon this formulation, **VAE-based prediction** extends the generative modeling process to sequential or dynamic data. Instead of reconstructing the same input, the model learns to predict *future states* by performing temporal modeling in the latent space. Specifically, given a latent encoding z_t obtained from the current observation x_t, a transition model $p_\psi(z_{t+1}|z_t)$ predicts the next-step latent representation, which is then decoded to generate the predicted state $\hat{x}_{t+1}$. This structure enables the model to capture uncertainty, multi-modal dynamics, and long-term dependencies that are difficult to model directly in the observation space. Consequently, VAE-based prediction has been widely applied in video prediction, motion forecasting, and physical simulation tasks, where learning a compact and stochastic latent representation facilitates accurate and physically plausible future-state generation.

A.2 Denoising Diffusion Probabilistic Models

Denoising Diffusion Probabilistic Models (DDPMs) are a class of generative models that learn to reverse a fixed forward noising process. During training, Gaussian noise is progressively added to the data in L steps, transforming structured samples into nearly pure noise. A neural network is then trained to de-noise these noisy samples step-by-step, learning to approximate the reverse of the diffusion process. At inference time, generation begins from random noise and proceeds iteratively using the learned reverse dynamics.

Formally, given a data distribution $q(x^0)$, the forward process adds Gaussian noise at each step:

$$q(x^l \mid x^{l-1}) = \mathcal{N}(x^l; \sqrt{1-\beta^l}x^{l-1}, \beta^l I), \tag{6}$$

where β^l is the noise schedule. The model is trained to approximate the reverse distribution $p_\theta(x^{l-1} \mid x^l)$ using a parameterized denoiser, often predicting either the original clean data x^0 or the added noise ϵ.

Diffusion models offer strong sample diversity and high-fidelity generation, making them well-suited for tasks that require modeling complex multimodal distributions. In the context of physical dynamics, their iterative refinement process is particularly advantageous, as it allows for structured generation that respects spatial and temporal coherence. In our setting, we adapt diffusion models to synthesize physically plausible trajectories given an initial state and a predicted final state, leveraging their flexibility to model diverse motion sequences (Ho et al., 2022; Blattmann et al., 2023).

B Details of ReStep-Driven Inverse Rendering

In this section, we describe how diffusion-generated particle trajectories are converted into updates on the Gaussian scene representation. Since different objects exhibit fundamentally different physical behaviors, we separately handle non-rigid (cloth-like) objects and rigid bodies.

B.1 Non-Rigid Objects

For deformable objects such as cloth, global rigid transformations are insufficient to capture local stretching and bending. We therefore update each Gaussian point using a locally interpolated displacement derived from the diffusion particle trajectories.

Let $P^{(0)}$ and $P^{(t)}$ denote the initial and the t-th diffusion-generated particle positions for the cloth. For a Gaussian point x, we first identify its k nearest cloth particles in the initial configuration. The displacement applied to x is then computed as a weighted average of the corresponding particle displacements:

$$\Delta x = \sum_{i=1}^{k} w_i \left(P_i^{(t)} - P_i^{(0)}\right), \qquad w_i = \frac{\exp(-\|x - P_i^{(0)}\|)}{\sum_j \exp(-\|x - P_j^{(0)}\|)}.$$

This formulation ensures that Gaussian points follow the local deformation field encoded by the diffusion particles while preserving smoothness and spatial coherence. Since non-rigid objects do not maintain a stable orientation, we only update point positions while keeping rotations unchanged.

B.2 Rigid Objects

Rigid bodies must preserve their shape, and therefore we determine a single global rigid transformation per object at each diffusion step. For an object, let $P^{(0)}$ and $P^{(t)}$ denote its initial and diffusion-predicted particle clouds. We compute the optimal rotation $R \in SO(3)$ and translation $t \in \mathbb{R}^3$ that align the initial cloud to the target cloud using the Kabsch algorithm:

$$R, t = \arg \min_{R \in SO(3),\, t} \sum_i \left\| P_i^{(t)} - (R P_i^{(0)} + t) \right\|^2.$$

The resulting rigid transform is applied to all Gaussian points belonging to the same object:

$$\hat{x} = Rx + t.$$

This ensures shape preservation, consistent motion, and physically meaningful dynamics for rigid objects.

B.3 Combined Update

At each diffusion step, the Gaussian scene is updated by combining the non-rigid and rigid motions:

$$X^{(t)} = X^{(t-1)} + \Delta X^{(t)}_{\text{nonrigid}} + \Delta X^{(t)}_{\text{rigid}}.$$

This hybrid formulation allows the Gaussian scene to faithfully follow diverse physical behaviors, supporting accurate reconstruction of both deformable materials and solid objects.

C Architecture Details

C.1 Stage 1: Stochastic Goal State Prediction

This section details the architectures of all sub-networks used in Stage 1. The network descriptions below correspond to Table 5, which summarizes the full architectural specifications and hyperparameters.

Let N denote the number of particles and $d = 3$ the spatial state dimension. Each particle is represented by its position and velocity. All graph-based modules operate on per-particle features and model local interactions using k-nearest-neighbor message passing. Unless otherwise specified, all MLPs use ReLU activations.

C.2 Stage 2: Anchored Diffusion

Anchored Diffusion is implemented with the Double-UNet architecture composed of multiple layers of convolution and attention operations. The network descriptions below accompany Table 6, which summarizes the full hyperparameter specifications. Each convolution block has two 1D convolution layers as its trunk net, with SiLU as its activation function. The diffusion step embedding is added to its intermediate feature. Each attention block has one self-attention layer, one cross-attention layer for condition injection, and one GeGLU layer for element-wise gating. Residual connection and group normalization are adopted for both blocks for stable training. Downsampling is implemented using a convolution layer with strided kernels, and upsampling is implemented using nearest neighbor interpolation.

D Additional Analyses and Experimental Diagnostics

D.1 Calibration of the Endpoint Predictor

We assess whether the uncertainty produced by the steady-state predictor is calibrated by measuring whether the ground-truth final state falls within nominal credible regions of the predicted distribution.

Table 7 reports the nominal confidence level p, the empirical coverage, and the average empirical threshold, defined as the physical distance from the predicted mean required to cover the ground truth. Coverage is close to the nominal value at higher confidence levels: it reaches 0.8800 at $p = 0.90$ and 0.9067 at $p = 0.95$. At $p = 0.50$, coverage is lower (0.3400), indicating under-coverage in more nonlinear contact-rich cases. The threshold increases monotonically with p, suggesting that the predicted uncertainty scales with the observed spatial variability.

Table 7: Calibration of the steady-state predictor on the Dominoes scene.

Nominal confidence p	Coverage	Threshold
0.50	0.3400	0.004806
0.80	0.7867	0.007224
0.90	0.8800	0.009154
0.95	0.9067	0.011467

D.2 Sensitivity to the Endpoint Condition

We compare three inputs to the trajectory generator: an endpoint predicted by the VAE, the observed endpoint (GT), and a perturbed endpoint (Random). The latter is constructed by adding clipped Gaussian perturbations to the initial condition. Table 8 reports trajectory MSE at two horizons. The GT condition is generally the most informative, whereas the predicted endpoint outperforms the perturbed control in all reported scenes. The predicted endpoint is not equivalent to the GT endpoint: the gap is particularly large on Roll at $t = 100$, and the VAE condition slightly improves on GT for Drop at $t = 50$. Thus, endpoint quality matters, but the observed ordering is not uniform across every scene and horizon.

Table 8: Trajectory MSE under different endpoint conditions (lower is better). Random endpoints are constructed by adding clipped Gaussian perturbations to the initial condition. The VAE condition uses a stochastic endpoint sample.

Scene	Endpoint condition	$t = 50$	$t = 100$
Dominoes	VAE	0.0017	0.0026
	GT	**0.0012**	**0.0014**
	Random	0.0047	0.0073
Drop	VAE	**0.0110**	0.0227
	GT	0.0122	**0.0174**
	Random	0.0400	0.0727
Roll	VAE	0.0500	0.0753
	GT	**0.0206**	**0.0211**
	Random	0.3083	0.4796

D.3 Endpoint-Predictor Backbone

Endpoint conditioning does not require a particular particle encoder. We replaced the GNN/k-NN endpoint predictor with a Transformer-style particle-set encoder and decoder, keeping the endpoint prediction objective. Table 9 shows comparable or slightly lower mean endpoint MSE for the Transformer in these scenes. We therefore do not attribute the benefit of endpoint conditioning to the GNN backbone; the GNN remains a locality-based choice for modeling particle interactions.

Table 9: Endpoint MSE (mean $\pm$ standard deviation; lower is better) for two endpoint-predictor backbones. These are endpoint prediction errors, not trajectory rollout errors.

Scene	GNN/k-NN	Transformer
Dominoes	0.0100 ± 0.0064	0.0093 ± 0.0074
Collide	0.3137 ± 1.2218	0.2946 ± 1.3407
Drop	0.0357 ± 0.0974	0.0357 ± 0.0928

D.4 Inference Cost and Particle-Count Scaling

Table 10 compares the reported inference costs of SGNN and the diffusion-based ReStep model. To compare a one-step autoregressive predictor with a full-trajectory generator, the reported FLOPs and time for ReStep are divided by trajectory length. On these two scenes, ReStep uses more FLOPs and parameters but has lower measured wall-clock time. This comparison reflects the evaluated implementations and hardware; FLOPs alone do not determine latency.

Table 10: Reported normalized inference cost per predicted step. ReStep's full-trajectory FLOPs and time are divided by trajectory length; SGNN predicts one step at a time. The parameter count is that of the compared models.

Scene	Model	FLOPs (B)	Time (s)	Parameters (M)
Dominoes	SGNN	63.254	1.258	1.551
	ReStep	121.742	0.055	5.36
Roll	SGNN	8.727	1.016	0.993
	ReStep	16.280	0.058	5.36

We separately profile the stage-1 GNN on scenes with different numbers of effective particles (Table 11). Each forward pass builds a $k = 16$ neighbor graph six times; static, non-informative large objects are excluded from the effective-particle counts. Within these evaluated scenes, total endpoint-predictor time ranges from 5.26 to 12.37 ms as the particle count ranges from 474 to 2028. This stage-1 microbenchmark is distinct from the full-model comparison in Table 10, and should not be read as full trajectory-generation latency.

Table 11: Stage-1 endpoint-predictor inference for different particle counts (batch size 1). Forward time excludes the separately measured k-NN construction; total time includes both.

Scene	Particles	FLOPs (B)	k-NN (ms)	Forward / total (ms)
Collide	361	14.81	1.53	4.05 / 5.58
Roll	474	19.45	1.49	3.77 / 5.26
Contain	603	24.75	1.52	4.02 / 5.53
Drop	1326	54.48	2.03	6.89 / 8.92
Link	1608	66.10	2.40	8.02 / 10.42
Support	2028	83.42	3.03	9.35 / 12.37

D.5 Endpoint Errors and Failure Cases

Endpoint errors can propagate into the completed trajectory. We examine 100 RigidFall samples and divide trajectory MSE into quartiles. Of the 25 samples with the smallest endpoint errors, 23 also fall into the lowest trajectory-error quartile. Of the 25 with the largest endpoint errors, 21 fall into the highest trajectory-error quartile. All ten samples above the 90th percentile of endpoint MSE fall into the highest trajectory-error quartile; their mean endpoint MSE is 0.2688, mean trajectory MSE is 0.1277, and mean trajectory MSE at $t = 100$ is 0.2488. These associations show that incorrect endpoints are a consequential failure mode. They do not, by themselves, establish how much of each trajectory error is caused by the endpoint error.

Table 12: Association between endpoint-error groups and trajectory-error quartiles on 100 RigidFall samples. Rows count samples in the lowest and highest endpoint-error quartiles; the remaining 50 samples are omitted.

Endpoint-error group	Traj. Q1	Traj. Q2	Traj. Q3	Traj. Q4
Lowest 25% ($n = 25$)	23	2	0	0
Highest 25% ($n = 25$)	0	0	4	21
Highest 10% ($n = 10$)	0	0	0	10

D.6 VAE-Based versus Deterministic Endpoint Prediction

To assess the benefit of the VAE-based endpoint predictor, we compare it with a deterministic endpoint predictor on the Contain and Drop scenes. As shown in Table 13, the VAE achieves lower mean endpoint MSE in both scenes, reducing the error from 0.4052 to 0.3731 on Contain and from 0.0484 to 0.0357 on Drop, corresponding to relative reductions of 7.9% and 26.2%, respectively. These results provide empirical support for the VAE-based formulation in endpoint prediction.

Table 13: Endpoint prediction errors for VAE-based and deterministic predictors on Contain and Drop. Lower mean endpoint MSE is highlighted in bold.

Scene	Endpoint predictor	Endpoint MSE ↓
Contain	VAE	$\mathbf{0.3731} \pm 0.5481$
	Deterministic model	0.4052 ± 0.5824
Drop	VAE	$\mathbf{0.0357} \pm 0.0974$
	Deterministic model	0.0484 ± 0.1151