# Dynamic 3D Gaussian Tracking for Graph-Based Neural Dynamics Modeling

**Mingtong Zhang**[1]*, **Kaifeng Zhang**[2]*, **Yunzhu Li**[2]
[1]University of Illinois Urbana-Champaign, [2]Columbia University

**Abstract:** Videos of robots interacting with objects encode rich information about the objects' dynamics. However, existing video prediction approaches typically do not explicitly account for the 3D information from videos, such as robot actions and objects' 3D states, limiting their use in real-world robotic applications. In this work, we introduce a framework to learn object dynamics directly from multi-view RGB videos by explicitly considering the robot's action trajectories and their effects on scene dynamics. We utilize the 3D Gaussian representation of 3D Gaussian Splatting (3DGS) to train a particle-based dynamics model using Graph Neural Networks. This model operates on sparse control particles downsampled from the densely tracked 3D Gaussian reconstructions. By learning the neural dynamics model on offline robot interaction data, our method can predict object motions under varying initial configurations and unseen robot actions. The 3D transformations of Gaussians can be interpolated from the motions of control particles, enabling the rendering of predicted future object states and achieving action-conditioned video prediction. The dynamics model can also be applied to model-based planning frameworks for object manipulation tasks. We conduct experiments on various kinds of deformable materials, including ropes, clothes, and stuffed animals, demonstrating our framework's ability to model complex shapes and dynamics. Our project page is available at https://gs-dynamics.github.io.

**Keywords:** Dynamics Model, 3D Gaussian Splatting, Action-Conditioned Video Prediction, Model-Based Planning

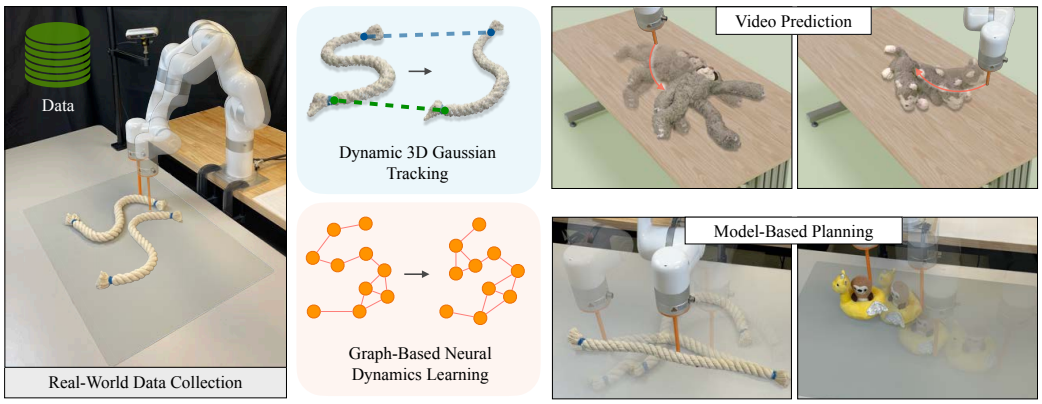


Figure 1: We propose a novel approach for learning a graph-based neural dynamics model from real-world data. By using videos captured from robot-object interactions, we achieve dense 3D tracking with a dynamic 3D Gaussian Splatting framework. Training a graph-based neural dynamics model on top of the 3D Gaussian particles enables action-conditioned video prediction and model-based planning.

*Equal contribution.

8th Conference on Robot Learning (CoRL 2024), Munich, Germany.

## 1 Introduction

Humans naturally grasp the dynamics of objects through observation and interaction, allowing them to intuitively predict how objects will move in response to specific actions [1]. This predictive capability enables humans to plan their behavior to achieve specific goals. In robotics, developing predictive models of object motions is critical for model-based planning frameworks [2–4]. However, learning such models directly from videos is challenging due to the complex physics involved in real-world object interactions. Previous works have sometimes used simulation environments to learn dynamics models, extracting ground truth trajectories of object states [5–7]. In contrast, real interaction data, such as videos, tend to be noisy and challenging to extract high-quality 3D information from, leading to inefficiencies in model learning.

Recent advancements in 3D Gaussian Splatting have introduced a novel approach for 3D reconstruction using a particle-based representation of a large number of 3D Gaussians trained with multiview images. Extending this to dynamic scenes allows for dense tracking by optimizing the temporal motions of these Gaussians. More importantly, these methods only fit the given videos and cannot predict object motions under unseen actions.

In this work, we introduce and enable 3D action-conditioned video prediction, a novel yet challenging capability in robotics. Achieving this necessitates both learning object dynamics and performing object reconstruction. The particle-based representation of Dyn3DGS [8] facilitates this by providing 3D correspondence in dynamic scenes while simultaneously reconstructing the appearance and geometry of objects. Utilizing the particle-based representation, we present a framework for learning predictive object dynamics from multi-view RGB videos. The framework has two main components: dynamic 3D Gaussians to obtain dense point correspondence in long-horizon robot-object interaction videos, and a neural dynamics model using Graph Neural Networks (GNNs) with optimized dense Gaussian correspondences as training data. Utilizing a spatial proximity graph from downsampled control particles, the model captures local dynamic relationships and generalizes well to unseen motions. By integrating 3D Gaussians with the particle-based dynamics model, our framework predicts dense Gaussian transformations through interpolating control point motions, enabling action-conditioned video prediction. The dynamics model can also be incorporated into a model-based planning pipeline for object manipulation, using model-predictive control to optimize actions for achieving target configurations.

We experiment on various objects, focusing on deformable materials like ropes, cloths, and toy animals. Results show our method accurately reconstructs 3D Gaussians and maintains coherent correspondences across frames, even with occlusions. With only 10 minutes of interaction video, we learn a dynamics model capturing distinct physical properties and generalizing well to unseen actions. Our method outperforms other system identification approaches, such as parameter optimization in physics-based simulators, in motion prediction accuracy and action-conditioned video prediction. We also demonstrate efficient model-based planning for object manipulations, showcasing the effectiveness of our framework in real-world dynamics learning.

## 2 Related Work

**Dense Correspondence from Videos.** Point tracking methods often rely on large datasets for 2D correspondences [9–12]. Tracking can be directly predicted from 2D videos, for instance, OmniMotion [13] enhances RAFT-predicted optical flows [14] using volumetric representations. Our approach is more related to another line of work, which focusing on lifting trackings from RGB or depth scans to the 3D space [15–17]. Recently, Dynamic 3D Gaussian Splatting approaches [8, 18] model the particles for effective tracking. Building on this line of work, our method uses 3D point scans for initialization and extends per-scene tracking to learn object dynamics, enabling motion prediction in unseen scenarios.

A primary incentive for examining point tracking is its potential impact on robotics applications [19, 20]. For instance, RoboTAP [21] demonstrates that pre-trained point-tracking models enhance the sample efficiency in visual imitation learning, and ATM [22] involves pre-training a

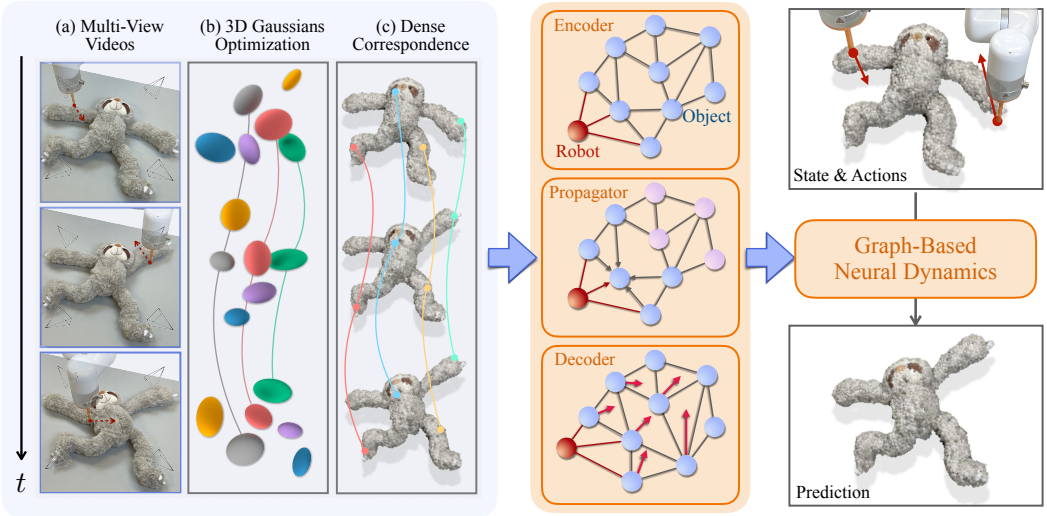


Figure 2: **Overview of Our Framework:** We first achieve dense 3D tracking of long-horizon robot-object interactions using multi-view videos and Dyn3DGS optimization. We then learn the object dynamics through a graph-based neural network. This approach enables applications such as (i) action-conditioned video prediction using linear blend skinning for motion prediction, and (ii) model-based planning for robotics.

model to predict future trajectories within a video frame. Our method learns neural dynamics model on top of particle-based tracking which enables efficient model-based planning.

**Neural Dynamics Modeling of Real-World Objects.** Modeling the dynamics of real-world objects is extremely challenging due to the high complexity of the state space and the variance of physical properties, especially for deformable materials. Using physics-based simulators, previous works have manually specified physical parameters [23], or performed gradient-free parameter search [24] and gradient-based methods [25–27] to optimize low-dimensional physical parameters in a continuous domain. However, these methods require accurate perception models to transfer objects into simulatable assets and thus are limited to constrained, manually designed environments. Neural network-based simulators can also learn object dynamics without requiring the analytical model [28–32]. Especially, graph-based networks can learn the dynamics of various different kinds of models such as plasticine [33, 34], cloth [6, 35], fluid [5, 36], etc. Our method builds upon these previous works but reduces the requirement of large-scale offline training data, enabling direct learning from videos, thus potentially eliminating sim-to-real gaps.

**Video Prediction for Robotics.** Autoregressive [37–39] and diffusion models [40–42] trained on Internet-scale data have demonstrated significant capabilities in video generation. Action-conditioned video prediction methods initially showed their capability in predicting future video frames conditioned on actions in simple scenarios, such as atari games [43, 44] and object push [45].

Recently, diffusion-based methods for video prediction [46–50] have created extended, realistic videos of robotic or human manipulations using large-scale data. Language-guided video prediction techniques [47, 51] use Vision-Language Models (VLMs) for extensive video planning sequences. These models can be extended to generating robotic policies by estimating object flow [52] or utilizing inverse dynamics [51]. Compared to these works, our work learns 3D action-conditioned video prediction from real-world data, which facilitates dynamics learning to support more complex robotic manipulation tasks.

# 3 Method

## 3.1 Preliminary: 3D Gaussian Splatting

3D Gaussian Splatting [53] optimizes a dense set of 3D Gaussians for explicit scene representation. Each Gaussian is defined by a 3D covariance matrix $\Sigma$ centered at $\mu$, represented as $G(x) = e^{-\frac{1}{2}(x)^T \Sigma^{-1}(x)}$. The matrix $\Sigma$ decomposes into a rotation matrix $R$ and a scale matrix $S$ ($\Sigma = RSS^T R^T$). For image rendering, 3D Gaussians are projected to 2D using a viewing trans-

formation $W$, with the camera coordinates covariance matrix $\Sigma'$ computed as $\Sigma' = JW\Sigma W^T J^T$, where $J$ is the Jacobian of the affine approximation of the projective transformation.

Scene representation optimization projects 3D Gaussians onto the image plane, converting their influence function into 2D for pixel evaluation. The image formation model [54] resembles point-based $\alpha$-blending and volumetric rendering, where pixel color $C$ is determined by blending multiple points: $C = \sum_{i \in N} c_i \alpha_i \prod_{j=1}^{i-1}(1 - \alpha_j)$. Here, $c_i$ is the color and $\alpha_i$ is the opacity value. The loss function for training is $\mathcal{L}_1$ with a D-SSIM term: $L = (1 - \lambda)\mathcal{L}_1 + \lambda \mathcal{L}_{\text{D-SSIM}}$, where $\lambda = 0.2$.

### 3.2 Dynamic 3D Gaussian Splatting with Dense Tracking

3D Gaussians are proven effective in modeling continuously-changing dynamic scenes. Dynamic 3D Gaussians (Dyn3DGS) [8] optimizes the spatial transformation of oriented Gaussian kernels over time, providing a dynamic view of scenes. We build upon Dyn3DGS for 6-DOF dense correspondence across timesteps, with the key insight that maintaining uniform Gaussian attributes–constant number, color, opacity, and size–while allowing position and orientation changes improve modeling accuracy and physical consistency. We use color-consistent ellipsoids to reduce optimization parameters and speed up training, which also empirically improves correspondence quality, especially under occlusions. Additionally, we only retain high-opacity Gaussians during training.

In line with Dyn3DGS, we use physics-inspired optimization objectives to guide optimization, ensuring physical plausibility and accurate long-term dense correspondence. The non-rigid physical modeling principles capture natural scene dynamics, enhancing the fidelity and stability of tracking over time. To overcome the challenges of our 4-view configuration, we strengthen the local rigidity and isometry objectives, improving the ability to handle limited observations effectively.

To efficiently isolate and optimize dynamic components, we use GroundingDINO [55] and Segment Anything [56] models to obtain masks for objects the robot interacts with. This allows us to focus on the Gaussians representing the objects of interest, enhancing optimization precision and efficiency.

### 3.3 Graph-Based Neural Dynamics Learning

The dense tracking of 3D Gaussians over a sequence of images provides a novel perspective in modeling the dynamics of objects during external interactions. Consider the scenario of an external robotic action applied to an object. The action can be represented as a sequence of end-effector positions $A = \{a_t\}_{0 \leq t \leq T}$. From the video sequence of the action, we can fit 3D Gaussians to the video using our modified Dyn3DGS, resulting in a collection of dynamic Gaussians: $X = \{X_t\}_{0 \leq t \leq T} = \{\mu_i^t\}_{1 \leq i \leq N, 0 \leq t \leq T}$, where $X_t$ denotes the set of Gaussians at time $t$, and $\mu_i^t$ denotes the position of a single Gaussian indexed $i$ at time $t$. The underlying physical dynamics function could then be depicted as

$$X_{t+1} = f(X_{0:t}, a_t) \tag{1}$$

for any timestep $t$. We can approximate this dynamics function with a neural network, parameterized by $\theta$. To model the relational inductive bias inherent in the continuum physics, we use a Graph Neural Network (GNN) architecture [57]. We apply the farthest point sampling on the Gaussians $X$ to extract a sparse subset of control particles $\hat{X} = \{\hat{X}_t\}_{0 \leq t \leq T} = \{\hat{\mu}_i^t\}_{1 \leq i \leq n, 0 \leq t \leq T}$, with $n$ particles and pairwise distance threshold $d_v$, and use them as graph vertices. The robot end-effector position $a_t$ is also considered as a graph vertex. A bidirectional edge is connected if two vertices' distance is below a threshold $d_e$, resulting in the edge set $\hat{E}$.

The vertex encoder takes as input the motion of each particle $\hat{\mu}_i^t - \hat{\mu}_i^{t-1}$ in the previous $k$ time steps ($k = 3$ in our experiments) and a one-hot vector indicating whether the vertex belongs to the object or the robot end-effector. To model the relationship between the object and table surface, the distance of every vertex to the surface is also included in the input. The edge encoder takes as input the 3-dimensional position difference of the two vertices it connects and a one-hot vector indicating whether the edge is an object-object relation or an object-robot relation.

The graph encodes vertices and edges into latent features using shared vertex and edge encoders, performs message passing for $p$ timesteps ($p = 3$ in our experiments), and uses a shared decoder to

map vertex latent features back to a 3-dimensional output indicating the motion at time $t$. The entire process can be represented as

$$\hat{X}_{t+1,pred} = \hat{X}_t + f_\theta(\hat{X}_{t-k:t}, \hat{E}_t). \tag{2}$$

**Training.** The main loss function to train our dynamics network is the MSE prediction error:

$$\mathcal{L}_{pred} = \sum_{i=1}^{\tau} \|\hat{X}_{t+i,pred} - \hat{X}_{t+i}\|^2, \tag{3}$$

where $\tau$ is the look-forward horizon size indicating how many steps we recurrently perform future prediction. We take $\tau = 5$ in our experiments to get a balance between prediction stability and computational cost. During training, we randomly sample time steps $t$ from the training sequences to serve as the starting frame.

Empirically, we also find adding regularization terms to be helpful in some objects with complex shapes. These include an edge length regularization term penalizing the mean squared difference of edge lengths between subsequent frames, and a rigidity regularization term, applied to objects that are rigid or close to rigid, penalizing the mean squared difference between the predicted motions and a rigid transformation estimated from the predictions [58].

### 3.4 Dense Motion Prediction on 3D Gaussians

The flexibility of 3D Gaussians allows us to perform motion densification from the sparse subset of particle $\hat{X}$ to the entire collection of Gaussians $X$ while maintaining the ability to render photo-realistic videos. Taking an initial static collection of Gaussians $X = \{X_0\}$ and the action sequence $A = \{a_t\}_{0 \le t \le T}$ as input, we first perform recurrent future prediction on the subsampled graph vertices, outputting $\hat{X}_{0:T}$. Then, for each timestep $t$, we perform the following interpolation scheme to derive the center motions and rotations of each Gaussian:

**Graph Vertex Transformations.** In the first step, we need to calculate the 6-DoF transformation for each graph vertex. The outputs of the GNN are per-vertex motions, which directly serve as the 3D translations. For the 3D rotation, for each vertex $\hat{\mu}_i^t$, we solve for the rotation $R_i^t$ that minimizes the error of its adjacent points in the graph:

$$R_i^t = \arg \min_{R \in SO(3)} \sum_{j \in \mathcal{N}(i)} \|R\hat{\mu}_j^t - \hat{\mu}_j^{t+1}\|^2. \tag{4}$$

**Gaussian Transformations.** We adopt Linear Blend Skinning (LBS) [59, 60] to calculate the motions of Gaussian centers. Concretely,

$$\mu_i^{t+1} = \sum_{b=1}^{n} w_{ib}^t R_b^t(\mu_i^t - \hat{\mu}_b^t) + \hat{\mu}_b^t + T_b^t, \quad q_i^{t+1} = \left(\sum_{b=1}^{n} w_{ib}^t r_b^t\right) \odot q_i^t, \tag{5}$$

where $w_{ib}^t$ is the weight associated between Gaussian $i$ and graph vertex $b$; $R_b^t, r_b^t$ are the matrix and quaternion representations of vertex $b$'s rotation at time $t$; $T_b^t$ is the translation of vertex $b$. The blending weight between a Gaussian and a vertex is inversely proportional to their 3D distance:

$$w_{ib}^t = \frac{\|\mu_j^t - \hat{\mu}_b^t\|^{-1}}{\sum_{b=1}^{n} \|\mu_j^t - \hat{\mu}_b^t\|^{-1}}, \tag{6}$$

thus assigning larger weights to vertices that are spatially closer to the Gaussian. We start from the initial Gaussian centers and rotations at $t = 0$ and repeatedly calculate the blending weights and updated Gaussian centers and rotations, resulting in dense Gaussian motions over the entire sequence.

### 3.5 Model-Based Planning

The dynamics network can also be used to perform model-based planning in robotics tasks. Given the multi-view RGB-D image of the object, we apply segmentation and point fusing to get the complete point cloud of the object. We then construct graph vertices and edges with FPS (Farthest Point

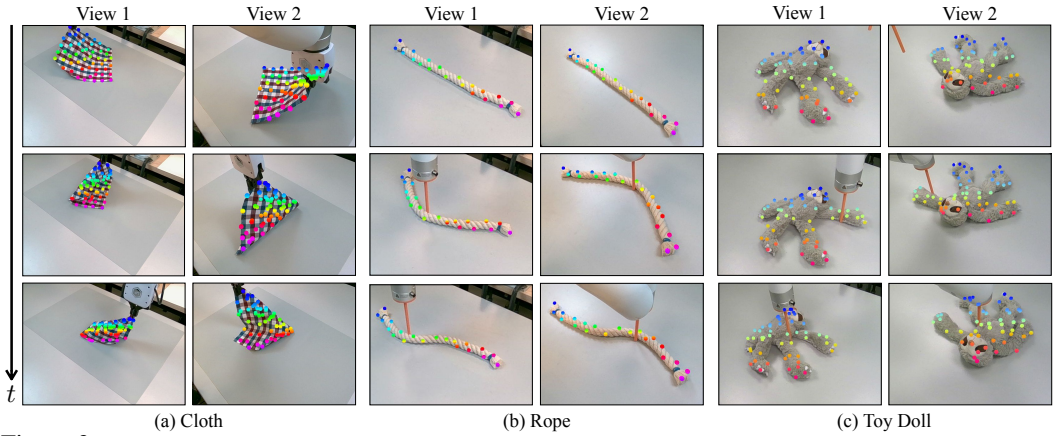

Figure 3: **Qualitative Results of 3D Gaussian Tracking.** We demonstrate point-level correspondence on the objects across various timesteps. Please check our website for more videos showcasing precise dense tracking even under different object deformations and occlusions.

Sampling). Given a target configuration, we use a Model Predictive Control (MPC) [61] framework to plan for robot actions. The optimization objective is defined as the mean squared difference between the target state and the predicted object state if an action is applied. In our experiments, we apply the Model-Predictive Path Integral (MPPI) [62] trajectory optimization algorithm.

# 4 Experiments

## 4.1 Experiment Setup

We perform experiments on 3D tracking, action-conditioned video prediction, and model-based planning. To achieve this, we collect a real-world dataset, with a focus on acquiring multiview images synchronized with corresponding actions at each timestep. The dataset consists of 3 types of deformable objects: rope, cloth, and toy animals, totaling 8 object instances. For all objects, we capture RGBD images and record robot actions at 15 FPS. The depth is used to initialize the point cloud for Gaussian representation. Our data collection setup includes four cameras, strategically positioned at the corners of the workspace and oriented to provide a comprehensive top-down perspective. This setup allows us to collect synchronized multiview data necessary for our experiments.

## 4.2 3D Tracking with Dynamic 3D Gaussians

We initialize Gaussian tracking with a point cloud from four views at the first timestep and optimize our Dyn3DGS for dense correspondence. To evaluate our method, we compare it with advanced 2D tracking approaches by projecting 4-view results into 3D using our depth data and evaluate the best results. The unmodified Dyn3DGS is considered as a baseline to show our improvements within the 4-view real-world experiment setup. Following prior work [63], we evaluate median trajectory error (MTE) in millimeters, position accuracy ($\delta$) at thresholds of 2, 4, 8, and 16 millimeters (reporting the average), and survival rate with a threshold of 0.5 meters. For a fair comparison, we also project our 3D tracking results into 2D pixels and evaluate the same metrics, allowing comprehensive assessment in both 2D and 3D contexts.

The quantitative results in Tab. 1 demonstrate that our 3D tracking method outperforms all baselines, including SpatialTracker [16], which uses depth as a 3D prior. The results highlight our method's capability across various scenes and objects. Additionally, the qualitative results in Fig. 3 show that our Dynamic 3D Gaussian tracking provides precise dense correspondence, even under challenging conditions such as occlusions from robot and object deformations.

## 4.3 Action-Conditioned Video Prediction

Using Dynamic 3D Gaussian reconstruction with tracking, we train graph-based neural dynamics models with downsampled control particles. These particle motions are used to interpolate dense

| Method | Metric | Rope | Cloth | Toy Animals | Metric | Rope | Cloth | Toy Animals |
|---|---|---|---|---|---|---|---|---|
| CoTracker [12] | 3D MTE [mm] ↓ | 46.96 | 55.84 | 51.84 | 2D MTE [mm] ↓ | 42.73 | 49.82 | 46.30 |
| PIPS++ [63] | | 49.37 | 58.10 | 58.45 | | 46.17 | 50.28 | 49.96 |
| SpatialTracker [16] | | 38.72 | 53.85 | 42.82 | | 32.29 | 46.26 | 37.49 |
| Dyn3DGS [8] | | 62.41 | 69.20 | 66.19 | | 57.39 | 61.28 | 60.17 |
| Ours | | **6.90** | **13.14** | **12.83** | | **4.92** | **11.72** | **10.94** |
| CoTracker [12] | 3D $\delta_{avg}$ ↑ | 79.28 | 75.79 | 75.46 | 2D $\delta_{avg}$ ↑ | 83.71 | 79.28 | 78.44 |
| PIPS++ [63] | | 69.83 | 66.92 | 68.95 | | 76.30 | 72.48 | 76.96 |
| SpatialTracker [16] | | 86.56 | 83.85 | 79.42 | | 92.36 | 90.52 | 89.62 |
| Dyn3DGS [8] | | 60.32 | 56.28 | 61.97 | | 67.20 | 62.28 | 67.96 |
| Ours | | **89.26** | **89.13** | **82.71** | | **93.27** | **92.18** | **94.19** |
| CoTracker [12] | 3D Survival ↑ | 94.41 | 95.19 | 92.26 | 2D Survival ↑ | 97.20 | **100** | 96.06 |
| PIPS++ [63] | | 92.61 | 91.39 | 85.73 | | 96.74 | 94.28 | 92.82 |
| SpatialTracker [16] | | **100** | 98.46 | 96.26 | | **100** | **100** | **100** |
| Dyn3DGS [8] | | 84.29 | 79.04 | 74.82 | | 87.83 | 82.14 | 79.32 |
| Ours | | **100** | **100** | **98.83** | | **100** | **100** | **100** |

Table 1: **Quantitative Results on Dynamic 3D Gaussian Tracking.** We labeled the ground truth for 200 frames per episode in 3D space for 1 or 2 object instances in each category, covering two episodes per object. Our Dyn3DGS-based tracking method outperforms all baselines, including the unmodified Dyn3DGS, in both 2D and 3D metrics.

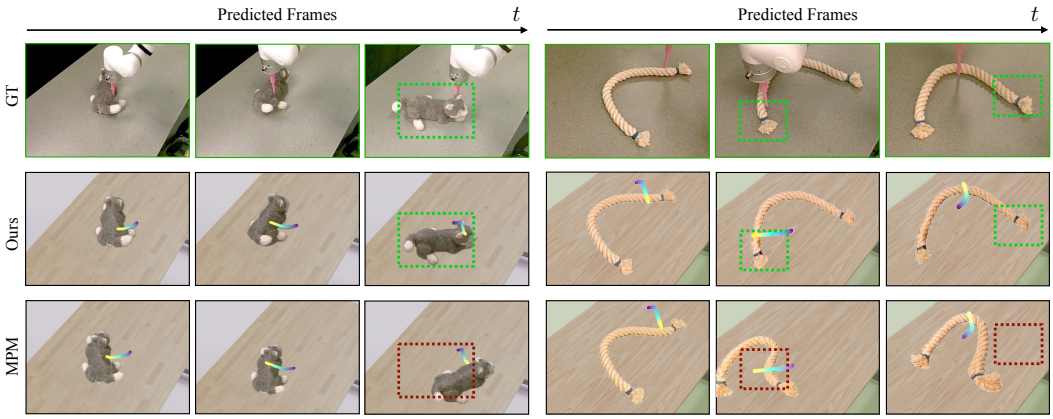


Figure 4: **Qualitative Results of Action-Conditioned 3D Video Prediction.** Our videos are generated by rendering predicted Gaussians on virtual backgrounds. Robot trajectories are visualized as curved lines (yellow: current end-effector positions, purple: history end-effector positions). Compared to the MPM baseline, our video prediction results align with the ground truth frames (GT) more accurately.

Gaussian kernel motion and rotations. To evaluate the quality of our dynamics prediction and 3D Gaussian rendering, we assess action-conditioned video prediction performance.

**Baselines.** Synthesizing action-conditioned object motions requires physics priors, and existing text-conditioned video prediction methods are not compatible with taking 3D robot actions as input. In this experiment, we consider 2 physics simulator-based baselines, *MPM* and *FleX*.

*MPM* is based on the Material Point Method simulation framework [64, 65]. Our baseline MPM uses the same simulation setting as previous works [23, 27], while adding support for two types of robot end-effectors: cylindrical pusher and gripper. For each object instance, we set the friction coefficient $\mu$ and Young's modulus $E$ (assumed uniform) to be two learnable parameters that are optimized from the training data using the CMA-ES algorithm. *FleX* is based on the NVIDIA FleX simulator. For all objects, we adopt the soft body simulator in FleX, reconstruct the initial object mesh from 3D Gaussians using alpha-shape mesh reconstruction. Similarly, we optimize the friction coefficient $\mu$ and stiffness coefficient $s$ (assumed uniform) of each object instance using CMA-ES.

**Metrics.** To evaluate the 3D video prediction performance, we adopt 3D distance metrics (3D Chamfer Distance, 3D Earth Mover's Distance), 2D segmentation similarity metrics ($\mathcal{J}\&\mathcal{F}$ score), and 2D image-based metrics (LPIPS). The metrics are complementary and together give a thorough comparison of future prediction accuracy and rendering quality.

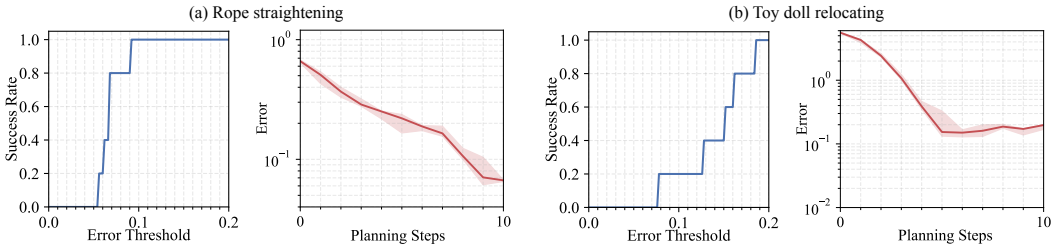


Figure 5: **Quantitative Results of model-based planning.** We perform each experiment 5 times and present the results as follows: (i) the median error curve relative to planning steps, with the area between 25 and 75 percentiles shaded, and (ii) the success rate curve relative to error thresholds.

**Results.** The results are listed in Tab. 2. Across all metrics, our method outperforms both baselines significantly, proving that we learn a more accurate dynamics model and give more realistic video prediction results. Our neural-based dynamics model effectively avoids the sim-to-real gap by learning from real-world data, thus improving the motion prediction accuracy under unseen robot actions and object configurations. This is also supported by the qualitative visualizations of predicted videos, shown in Fig. 4.

| Method | Metric | Cloth | Toy Animals | Rope | Metric | Cloth | Toy Animals | Rope |
|---|---|---|---|---|---|---|---|---|
| MPM | 3D Chamfer ↓ | 0.076 | 0.052 | 0.073 | $\mathcal{J}$ score↑ | 0.421 | 0.602 | 0.397 |
| FleX | | 0.136 | 0.069 | 0.074 | | 0.268 | 0.451 | 0.299 |
| Ours | | **0.064** | **0.030** | **0.053** | | **0.538** | **0.701** | **0.419** |
| MPM | 3D EMD ↓ | 0.097 | 0.053 | 0.066 | $\mathcal{F}$ score ↑ | 0.296 | 0.598 | 0.600 |
| FleX | | 0.126 | 0.066 | 0.068 | | 0.182 | 0.424 | 0.552 |
| Ours | | **0.067** | **0.035** | **0.048** | | **0.446** | **0.726** | **0.667** |
| MPM | LPIPS ↓ | 0.057 | 0.027 | 0.030 | $\mathcal{J}\&\mathcal{F}$ score ↑ | 0.359 | 0.600 | 0.498 |
| FleX | | N/A | N/A | N/A | | 0.225 | 0.438 | 0.426 |
| Ours | | **0.044** | **0.021** | **0.025** | | **0.492** | **0.714** | **0.543** |

Table 2: **Quantitative Results on Action-Conditioned 3D Video Prediction.** We evaluate a test sequence set for each object instance and present the results averaged by object category. Our method outperforms simulator baselines across all metrics.

### 4.4 Model-Based Planning

The action-conditioned dynamics model enables deployment in real-world robot planning tasks, demonstrating its generalizability to unseen object configurations. We choose rope straightening and toy animal relocating tasks to show our model's ability to manipulate highly deformable objects to target configurations. The quantitative results in Fig. 5 demonstrate that our approach effectively reduces error and achieves high success rates. The computational complexity and inefficiency of MPM and FleX baselines make them impractical to perform real-world planning tasks.

## 5 Conclusion and Limitation

In this paper, we introduce a novel approach for graph-based neural dynamics modeling from real-world data for 3D action-conditioned video prediction and planning. Our method reconstructs 3D Gaussians with cross-frame correspondences and learns a neural dynamics model that facilitates action-conditioned future prediction. The method outperforms baseline approaches in motion prediction and video prediction accuracy. We also showcase our model-based planning performance for object manipulation tasks, highlighting our framework's effectiveness in real-world robotic applications.

**Limitation** Although our approach learns directly from videos, collecting real-world data is still costly, and the perception module may fail when there are large occlusions or textureless objects and. In future work, we aim to develop a more efficient data collection pipeline and a more robust perception system to reduce the learning cost and enhance the method's applicability to all kinds of real videos.

**Acknowledgments**

The Toyota Research Institute (TRI) partially supported this work. This article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity. We would like to thank Haozhe Chen, Binghao Huang, Hanxiao Jiang, Yixuan Wang and Ruihai Wu for their thoughtful discussions. We thank Binghao Huang and Hanxiao Jiang for their help on the real-world data collection framework design, and also thank Binghao Huang and Haozhe Chen for their help with visualizations.

## References

[1] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40:e253, 2017.

[2] C. Finn and S. Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE, 2017.

[3] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*, pages 1101–1112. PMLR, 2020.

[4] L. Manuelli, Y. Li, P. Florence, and R. Tedrake. Keypoints into the future: Self-supervised correspondence in model-based reinforcement learning. *arXiv preprint arXiv:2009.05085*, 2020.

[5] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pages 8459–8468. PMLR, 2020.

[6] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020.

[7] Y. Wang, Y. Li, K. Driggs-Campbell, L. Fei-Fei, and J. Wu. Dynamic-resolution model learning for object pile manipulation. *arXiv preprint arXiv:2306.16700*, 2023.

[8] J. Luiten, G. Kopanas, B. Leibe, and D. Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, 2024.

[9] C. Doersch, A. Gupta, L. Markeeva, A. Recasens, L. Smaira, Y. Aytar, J. Carreira, A. Zisserman, and Y. Yang. Tap-vid: A benchmark for tracking any point in a video. *Advances in Neural Information Processing Systems*, 35:13610–13626, 2022.

[10] C. Doersch, Y. Yang, M. Vecerik, D. Gokay, A. Gupta, Y. Aytar, J. Carreira, and A. Zisserman. Tapir: Tracking any point with per-frame initialization and temporal refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10061–10072, 2023.

[11] A. W. Harley, Z. Fang, and K. Fragkiadaki. Particle video revisited: Tracking through occlusions using point trajectories. In *European Conference on Computer Vision*, pages 59–75. Springer, 2022.

[12] N. Karaev, I. Rocco, B. Graham, N. Neverova, A. Vedaldi, and C. Rupprecht. Cotracker: It is better to track together. *arXiv preprint arXiv:2307.07635*, 2023.

[13] Q. Wang, Y.-Y. Chang, R. Cai, Z. Li, B. Hariharan, A. Holynski, and N. Snavely. Tracking everything everywhere all at once. In *International Conference on Computer Vision*, 2023.

[14] Z. Teed and J. Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 402–419. Springer, 2020.

[15] R. A. Newcombe, D. Fox, and S. M. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 343–352, Los Alamitos, CA, USA, jun 2015. IEEE Computer Society. doi:10.1109/CVPR.2015.7298631. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2015.7298631.

[16] Y. Xiao, Q. Wang, S. Zhang, N. Xue, S. Peng, Y. Shen, and X. Zhou. Spatialtracker: Tracking any 2d pixels in 3d space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[17] C. Wang, B. Eckart, S. Lucey, and O. Gallo. Neural trajectory fields for dynamic novel view synthesis. *arXiv preprint arXiv:2105.05994*, 2021.

[18] B. P. Duisterhof, Z. Mandi, Y. Yao, J.-W. Liu, M. Z. Shou, S. Song, and J. Ichnowski. Md-splatting: Learning metric deformation from 4d gaussians in highly deformable scenes. *arXiv preprint arXiv:2312.00583*, 2023.

[19] H. Bharadhwaj, R. Mottaghi, A. Gupta, and S. Tulsiani. Track2act: Predicting point tracks from internet videos enables diverse zero-shot robot manipulation, 2024.

[20] M. Xu, Z. Xu, Y. Xu, C. Chi, G. Wetzstein, M. Veloso, and S. Song. Flow as the cross-domain manipulation interface. *arXiv preprint arXiv:2407.15208*, 2024.

[21] M. Vecerik, C. Doersch, Y. Yang, T. Davchev, Y. Aytar, G. Zhou, R. Hadsell, L. Agapito, and J. Scholz. Robotap: Tracking arbitrary points for few-shot visual imitation. *arXiv*, 2023.

[22] C. Wen, X. Lin, J. So, K. Chen, Q. Dou, Y. Gao, and P. Abbeel. Any-point trajectory modeling for policy learning, 2023.

[23] T. Xie, Z. Zong, Y. Qiu, X. Li, Y. Feng, Y. Yang, and C. Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. *arXiv preprint arXiv:2311.12198*, 2023.

[24] B. Wang, L. Wu, K. Yin, U. M. Ascher, L. Liu, and H. Huang. Deformation capture and modeling of soft objects. *ACM Trans. Graph.*, 34(4):94–1, 2015.

[25] Y.-L. Qiao, J. Liang, V. Koltun, and M. C. Lin. Differentiable simulation of soft multi-body systems. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[26] X. Li, Y.-L. Qiao, P. Y. Chen, K. M. Jatavallabhula, M. Lin, C. Jiang, and C. Gan. Pac-nerf: Physics augmented continuum neural radiance fields for geometry-agnostic system identification. *arXiv preprint arXiv:2303.05512*, 2023.

[27] T. Zhang, H.-X. Yu, R. Wu, B. Y. Feng, C. Zheng, N. Snavely, J. Wu, and W. T. Freeman. PhysDreamer: Physics-based interaction with 3d objects via video generation. *arxiv*, 2024.

[28] Y. Wu, W. Yan, T. Kurutach, L. Pinto, and P. Abbeel. Learning to manipulate deformable objects without demonstrations. *arXiv preprint arXiv:1910.13439*, 2019.

[29] P. Ma, P. Y. Chen, B. Deng, J. B. Tenenbaum, T. Du, C. Gan, and W. Matusik. Learning neural constitutive laws from motion observations for generalizable pde dynamics. In *International Conference on Machine Learning*, pages 23279–23300. PMLR, 2023.

[30] Z. Xu, J. Wu, A. Zeng, J. B. Tenenbaum, and S. Song. Densephysnet: Learning dense physical object representations via multi-step dynamic interactions. *arXiv preprint arXiv:1906.03853*, 2019.

[31] B. Evans, A. Thankaraj, and L. Pinto. Context is everything: Implicit identification for dynamics adaptation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2642–2648. IEEE, 2022.

[32] Z. Chen, K. Yi, Y. Li, M. Ding, A. Torralba, J. B. Tenenbaum, and C. Gan. Comphy: Compositional physical reasoning of objects and events from videos. *arXiv preprint arXiv:2205.01089*, 2022.

[33] H. Shi, H. Xu, Z. Huang, Y. Li, and J. Wu. Robocraft: Learning to see, simulate, and shape elasto-plastic objects in 3d with graph networks. *The International Journal of Robotics Research*, 0(0):02783649231219020, 0. doi:10.1177/02783649231219020. URL https://doi.org/10.1177/02783649231219020.

[34] H. Shi, H. Xu, S. Clarke, Y. Li, and J. Wu. Robocook: Long-horizon elasto-plastic object manipulation with diverse tools. *arXiv preprint arXiv:2306.14447*, 2023.

[35] X. Lin, Y. Wang, Z. Huang, and D. Held. Learning visible connectivity dynamics for cloth smoothing. In *Conference on Robot Learning*, pages 256–266. PMLR, 2022.

[36] Y. Li, J. Wu, R. Tedrake, J. B. Tenenbaum, and A. Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *ICLR*, 2019.

[37] D. Kondratyuk, L. Yu, X. Gu, J. Lezama, J. Huang, R. Hornung, H. Adam, H. Akbari, Y. Alon, V. Birodkar, et al. Videopoet: A large language model for zero-shot video generation. *arXiv preprint arXiv:2312.14125*, 2023.

[38] R. Villegas, M. Babaeizadeh, P.-J. Kindermans, H. Moraldo, H. Zhang, M. T. Saffar, S. Castro, J. Kunze, and D. Erhan. Phenaki: Variable length video generation from open domain textual descriptions. In *International Conference on Learning Representations*, 2022.

[39] C. Wu, J. Liang, L. Ji, F. Yang, Y. Fang, D. Jiang, and N. Duan. Nüwa: Visual synthesis pre-training for neural visual world creation. In *European conference on computer vision*, pages 720–736. Springer, 2022.

[40] A. Blattmann, T. Dockhorn, S. Kulal, D. Mendelevitch, M. Kilian, D. Lorenz, Y. Levi, Z. English, V. Voleti, A. Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

[41] J. Ho, W. Chan, C. Saharia, J. Whang, R. Gao, A. Gritsenko, D. P. Kingma, B. Poole, M. Norouzi, D. J. Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

[42] T. Brooks, B. Peebles, C. Homes, W. DePue, Y. Guo, L. Jing, D. Schnurr, J. Taylor, T. Luhman, E. Luhman, et al. Video generation models as world simulators, 2024.

[43] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh. Action-conditional video prediction using deep networks in atari games. *Advances in neural information processing systems*, 28, 2015.

[44] C. Finn, I. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. *Advances in neural information processing systems*, 29, 2016.

[45] H. Yuan, R. Wu, A. Zhao, H. Zhang, Z. Ding, and H. Dong. Dmotion: Robotic visuomotor control with unsupervised forward model learned from videos. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7135–7142. IEEE, 2021.

[46] M. Yang, Y. Du, K. Ghasemipour, J. Tompson, D. Schuurmans, and P. Abbeel. Learning interactive real-world simulators. *arXiv preprint arXiv:2310.06114*, 2023.

[47] Y. Du, S. Yang, P. Florence, F. Xia, A. Wahid, P. Sermanet, T. Yu, P. Abbeel, J. B. Tenenbaum, L. P. Kaelbling, et al. Video language planning. In *The Twelfth International Conference on Learning Representations*, 2023.

[48] S. Yang, J. Walker, J. Parker-Holder, Y. Du, J. Bruce, A. Barreto, P. Abbeel, and D. Schuurmans. Video as the new language for real-world decision making. *ICML*, 2024.

[49] J. Liang, R. Liu, E. Ozguroglu, S. Sudhakar, A. Dave, P. Tokmakov, S. Song, and C. Vondrick. Dreamitate: Real-world visuomotor policy learning via video generation, 2024.

[50] B. Van Hoorick, R. Wu, E. Ozguroglu, K. Sargent, R. Liu, P. Tokmakov, A. Dave, C. Zheng, and C. Vondrick. Generative camera dolly: Extreme monocular dynamic novel view synthesis. 2024.

[51] Y. Du, S. Yang, B. Dai, H. Dai, O. Nachum, J. Tenenbaum, D. Schuurmans, and P. Abbeel. Learning universal policies via text-guided video generation. *Advances in Neural Information Processing Systems*, 36, 2024.

[52] P.-C. Ko, J. Mao, Y. Du, S.-H. Sun, and J. B. Tenenbaum. Learning to Act from Actionless Video through Dense Correspondences. *arXiv:2310.08576*, 2023.

[53] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023. URL https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/.

[54] G. Kopanas, T. Leimkühler, G. Rainer, C. Jambon, and G. Drettakis. Neural point catacaustics for novel-view synthesis of reflections. *ACM Transactions on Graphics*, 41(6):Article–201, 2022.

[55] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, C. Li, J. Yang, H. Su, J. Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023.

[56] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

[57] Y. Li, J. Wu, J.-Y. Zhu, J. B. Tenenbaum, A. Torralba, and R. Tedrake. Propagation networks for model-based control under partial observation. In *ICRA*, 2019.

[58] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 13(04):376–380, 1991.

[59] R. W. Sumner, J. Schmid, and M. Pauly. Embedded deformation for shape manipulation. *ACM Trans. Graph.*, 26(3):80–es, jul 2007. ISSN 0730-0301. doi:10.1145/1276377.1276478. URL https://doi.org/10.1145/1276377.1276478.

[60] Y.-H. Huang, Y.-T. Sun, Z. Yang, X. Lyu, Y.-P. Cao, and X. Qi. Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. *arXiv preprint arXiv:2312.14937*, 2023.

[61] E. Camacho and C. Alba. *Model Predictive Control*. Advanced Textbooks in Control and Signal Processing. Springer London, 2013. ISBN 9780857293985. URL https://books.google.com/books?id=tXZDAAAAQBAJ.

[62] G. Williams, A. Aldrich, and E. A. Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357, 2017.

[63] Y. Zheng, A. W. Harley, B. Shen, G. Wetzstein, and L. J. Guibas. Pointodyssey: A large-scale synthetic dataset for long-term point tracking. In *ICCV*, 2023.

[64] D. Sulsky, S.-J. Zhou, and H. L. Schreyer. Application of a particle-in-cell method to solid mechanics. *Computer physics communications*, 87(1-2):236–252, 1995.

[65] Y. Hu, Y. Fang, Z. Ge, Z. Qu, Y. Zhu, A. Pradhana, and C. Jiang. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.