PhysTwin: Physics-Informed Reconstruction and Simulation of Deformable Objects from Videos

Hanxiao Jiang[1,2] Hao-Yu Hsu[2] Kaifeng Zhang[1] Hsin-Ni Yu[2] Shenlong Wang[2] Yunzhu Li[1]
[1]Columbia University [2]University of Illinois Urbana-Champaign

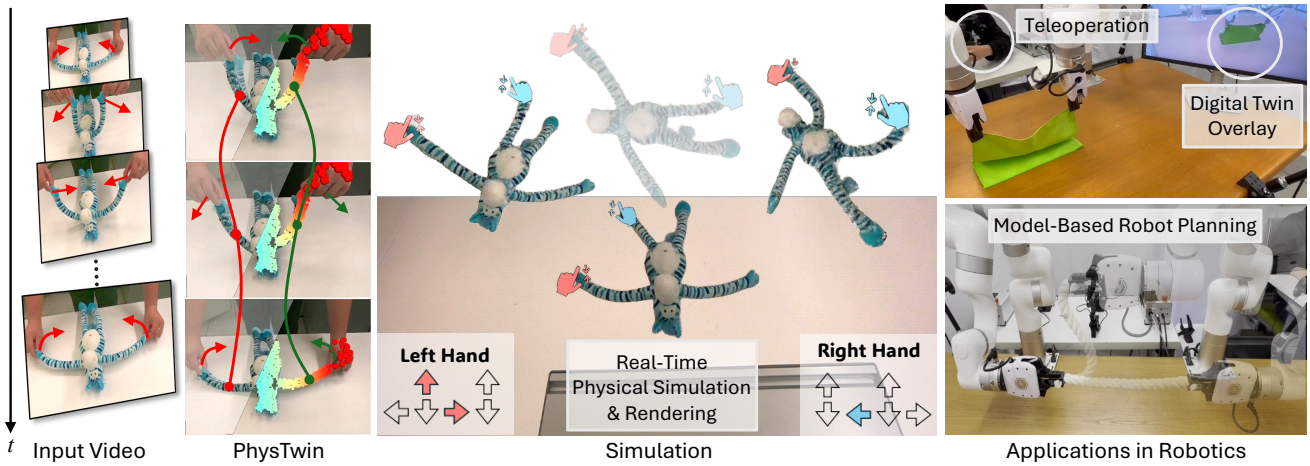


Figure 1. **PhysTwin** takes sparse videos (three camera views) of deformable objects under interaction as input and reconstructs a simulatable digital twin with complete geometry, high-fidelity appearance, and accurate physical parameters. This enables multiple applications, such as real-time interactive simulation using keyboards and robotic teleoperation devices, as well as model-based robot planning.

Abstract

Creating a physical digital twin of a real-world object has immense potential in robotics, content creation, and XR. In this paper, we present PhysTwin, a novel framework that uses sparse videos of dynamic objects under interaction to produce a photo- and physically realistic, real-time interactive virtual replica. Our approach centers on two key components: (1) a physics-informed representation that combines spring-mass models for realistic physical simulation, generative shape models for geometry, and Gaussian splats for rendering; and (2) a novel multi-stage, optimization-based inverse modeling framework that reconstructs complete geometry, infers dense physical properties, and replicates realistic appearance from videos. Our method integrates an inverse physics framework with visual perception cues, enabling high-fidelity reconstruction even from partial, occluded, and limited viewpoints. PhysTwin supports modeling various deformable objects, including ropes, stuffed animals, cloth, and delivery packages. Experiments show that PhysTwin outperforms competing methods in reconstruction, rendering, future prediction, and simulation under novel interactions. We further demonstrate its applications in interactive real-time simulation and model-based robotic motion planning. Project Page: https://jianghanxiao.github.io/phystwin-web/

1. Introduction

The construction of interactive digital twins is essential for modeling the world and simulating future states, with applications in virtual reality, augmented reality, and robotic manipulation. A physically realistic digital twin (PhysTwin) should accurately capture the geometry, appearance, and physical properties of an object, allowing simulations that closely match observations in the real world. However, constructing such a representation from sparse observations remains a significant challenge.

The creation of digital twins for deformable objects has long been a challenging topic in the vision community. While dynamic 3D methods (e.g., dynamic NeRFs [2, 5, 8, 13, 14, 17, 27, 29–31, 39–41, 43, 55, 56, 58, 61], dynamic 3D Gaussians [10, 20, 24, 33, 34, 59, 65, 66, 68]) capture observed motion, appearance, and geometry from videos, they omit the underlying physics and are thus unsuitable for simulating outcomes in unseen interactions. While recent neural-based models [4, 11, 28, 32, 36, 42, 49, 51, 52, 60, 64, 69] learn intuitive physics models from videos, they require large amounts of data and remain limited to specific objects or motions, whereas physics-driven approaches [9, 12, 27, 44, 63, 71, 72] often rely on pre-scanned shapes or dense observations to mitigate ill-posedness. Additionally, it requires dense viewpoint coverage and supports only limited motion

types, making it unsuitable for general dynamics modeling.

In this work, we aim to build an interactive PhysTwin from sparse-viewpoint RGB-D video sequences, capturing object geometry, non-rigid dynamic physics, and appearance for realistic physical simulation and rendering. We model deformable object dynamics with a spring-mass-based representation, enabling efficient physical simulation and handling a wide range of common objects, such as ropes, stuffed animals, cloth, and delivery packages. To address the challenges posed by sparse observations, we leverage shape priors and motion estimation from advanced 3D generative models [62] and vision foundation models [23, 46, 48] to estimate the topology, geometry, and physical parameters of our physical representation. Since some physical parameters (such as topology-related properties) are non-differentiable and optimizing them efficiently is non-trivial, we design a hierarchical sparse-to-dense optimization strategy. This strategy integrates zero-order optimization [18] for non-differentiable topology and sparse physical parameters (e.g., collision parameters and homogeneous spring stiffness), while employing first-order gradient-based optimization to refine dense spring stiffness and further optimize collision parameters. For appearance modeling, we adopt a Gaussian blending strategy, initializing static Gaussians from sparse observations in the first frame using shape priors and deforming them with a linear blending algorithm to generate realistic dynamic appearances.

Our inverse modeling framework effectively constructs interactive PhysTwin from videos of objects under interaction. We create a real-world deformable object interaction dataset and evaluate our method on three key tasks: reconstruction and resimulation, future prediction, and generalization to unseen interactions. Both quantitative and qualitative results demonstrate that our reconstructed PhysTwin aligns accurately with real-world observations, achieves precise future predictions, and generates realistic simulations under diverse unseen interactions. Furthermore, the high computational efficiency of our physics simulator enables real-time dynamics and rendering of our constructed PhysTwin, facilitating multiple applications, including real-time interactive simulation and model-based robotic motion planning.

2. Related Works

Dynamic Scene Reconstruction. Dynamic scene reconstruction aims to recover the underlying representation of dynamic scenes from inputs like depth scans [6, 26], RGBD videos [38], or monocular or multi-view videos [1, 5, 24, 31, 34, 39, 40, 43, 56, 58, 61, 67, 68]. Recent advancements in dynamic scene modeling have involved the adaptation of novel scene representations, including Neural Radiance Fields (NeRF) [2, 5, 8, 13, 14, 16, 17, 27, 29, 30, 30, 31, 39–41, 43, 55, 56, 58, 61] and 3D Gaussian splats [10, 20, 24, 33, 34, 59, 65, 66, 68]. D-NeRF [43] extends a canonical NeRF on dynamic scenes by optimizing a deformable field. Similarly, Deformable 3D-GS [66] optimizes a deformation field of each Gaussian kernel. Dynamic 3D-GS [34] optimizes the motion of Gaussian kernels for each frame to capture scene dynamics. 4D-GS [59] modulates 3D Gaussians with 4D neural voxels for dynamic multi-view synthesis. Although these methods achieve high-fidelity results in dynamic multi-view synthesis, they primarily focus on reconstructing scene appearance and geometry without capturing real-world dynamics, limiting their ability to support action-conditioned future predictions and interactive simulations.

Physics-Based Simulation of Deformable Objects. Another line of work incorporates physical simulators to perform system identification of physical parameters during reconstruction. Earlier methods relied on pre-scanned static objects and required clean point cloud observations [9, 15, 19, 21, 35, 44, 47, 57]. Most recent approaches build upon SDF [45], NeRF [3, 12, 27] or Gaussian Splatting [22, 63, 71, 72] to support more flexible physical digital twin reconstruction. Several works [12, 22, 63] manually specify physics parameters, resulting in a mismatch between the simulation and real-world video observations. Other works [3, 27, 45, 71, 72] attempt to estimate physical parameters from videos. However, they are often constrained to synthetic data, limited motion, or the need for dense viewpoints to accurately reconstruct static geometry, limiting their practical applicability. The closest related work to ours is Spring-Gaus [72], which also utilizes a 3D Spring-Mass model for learning from videos. However, their physical model is overly regularized and violates real-world physics, lacking momentum conservation and realistic gravity. Moreover, Spring-Gaus requires dense viewpoint coverage to reconstruct the full geometry at the initial state, which is impractical in many real-world settings. The motions are also limited to tabletop collisions and lack action inputs, making Spring-Gaus unsuitable as a general dynamics model for downstream applications.

Learning-Based Simulation of Deformable Objects. Analytically modeling the dynamics of deformable objects is challenging due to the high complexity of the state space and the variability of physical properties. Recent works [4, 11, 36, 60, 64] have chosen to use neural network-based simulators to model object dynamics. Specifically, graph-based networks effectively learn the dynamics of various types of objects such as plasticine [51, 52], cloth [32, 42], fluid [28, 49], and stuffed animals [69]. GS-Dynamics [69] attempted to learn object dynamics directly from real-world videos using tracking and appearance priors from Dynamic Gaussians [34], and generalized well to unseen actions. However, these learned models need extensive training samples and are often limited to specific environments with limited motion ranges. In contrast, our method requires only one interaction trial while achieving a broader range of motions.

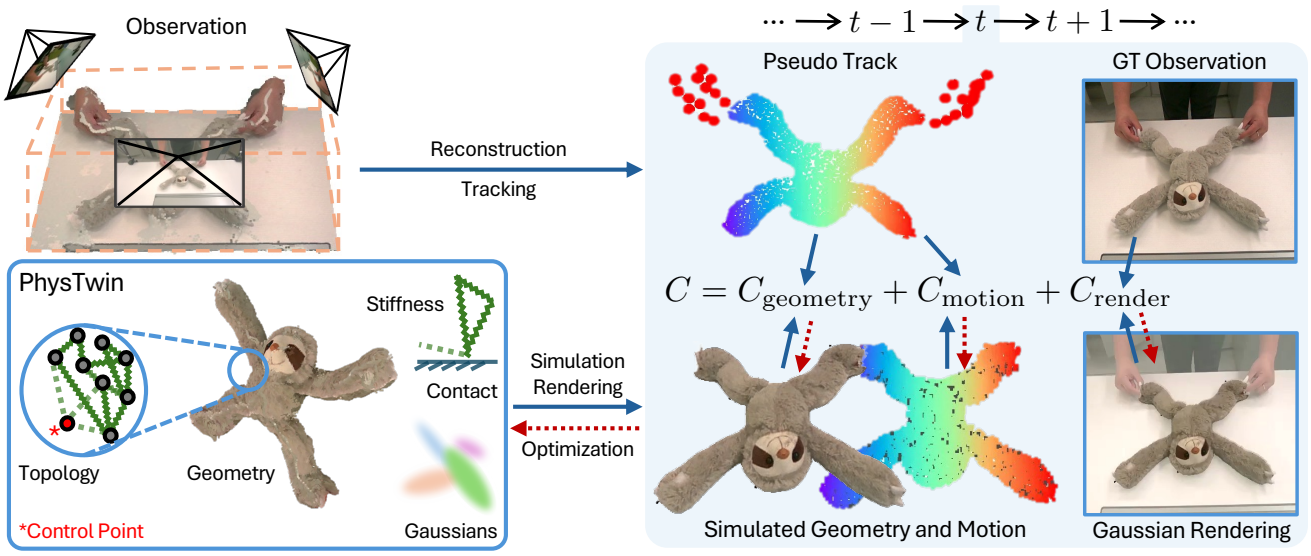


Figure 2. **Overview of Our PhysTwin Framework.** We present an overview of our PhysTwin framework, where the core representation includes geometry, topology, physical parameters (associated with springs and contacts), and Gaussian kernels. To optimize PhysTwin, we minimize the rendering loss and the discrepancy between simulated and observed geometry/motion. The rendering loss optimizes the Gaussian kernels, while the geometry and motion losses refine the overall geometry, topology, and physical parameters in PhysTwin.

3. Preliminary: Spring-Mass Model

Spring-mass models are widely used for simulating deformable objects due to their simplicity and computational efficiency. A deformable object is represented as a set of spring-connected mass nodes, forming a graph structure $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of mass points and $\mathcal{E}$ is the set of springs. Each mass node i has a position $\mathbf{x}_i \in \mathbb{R}^3$ and velocity $\mathbf{v}_i \in \mathbb{R}^3$, which evolve over time according to Newtonian dynamics. Springs are constructed between neighboring nodes based on a predefined topology, defining the elastic structure of the object.

The force on node i is the result of the combined effects of adjacent nodes connected by springs:

$$\mathbf{F}_i = \sum_{(i,j)\in\mathcal{E}} \mathbf{F}_{i,j}^{\text{spring}} + \mathbf{F}_{i,j}^{\text{dashpot}} + \mathbf{F}_i^{\text{ext}}, \tag{1}$$

where the spring force and dashpot damping force between nodes i and j are given by $\mathbf{F}_{i,j}^{\text{spring}} = k_{ij}(\|\mathbf{x}_j - \mathbf{x}_i\| - l_{ij})\frac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|}$ and $\mathbf{F}_{i,j}^{\text{dashpot}} = -\gamma(\mathbf{v}_i - \mathbf{v}_j)$, respectively. Here, k_{ij} is the spring stiffness, l_{ij} is the rest length, and γ is the dashpot damping coefficient. The external force $\mathbf{F}_i^{\text{ext}}$ accounts for factors such as gravity, collisions, and user interactions. The spring force restores the system to its rest shape, while the dashpot damping dissipates energy, preventing oscillations. For collisions, we use impulse-based collision handling when two mass points are very close, including collisions between the object and the collider, as well as between two object points.

The spring-mass model updates the system state with a dynamic model $\mathbf{X}_{t+1} = f_{\alpha,\mathcal{G}_0}(\mathbf{X}_t, a_t)$ by applying explicit Euler integration to both velocity and position. More formally, for all i, $\mathbf{v}_i^{t+1} = \delta\left(\mathbf{v}_i^t + \Delta t \frac{\mathbf{F}_i}{m_i}\right)$, $\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \Delta t\, \mathbf{v}_i^{t+1}$, where $\mathbf{X}_t$ represents the system state at time t, and δ represents the drag damping. In this formulation, α denotes all physical parameters of the spring-mass model, including spring stiffness, collision parameters, and damping. It also encompasses the parameters related to the control interaction. $\mathcal{G}_0$ represents the "canonical" geometry and topology for the spring-mass system[1], and a_t represents the action at time t.

4. Method

In this section, we formulate the construction of PhysTwin as an optimization problem. We then present our two-stage strategy, where the first stage addresses the physics-related optimization, followed by the appearance-based optimization in the second stage. Finally, we demonstrate the capability of our framework to perform real-time simulation using the constructed PhysTwin.

4.1. Problem Formulation

Given three RGBD videos of a deformable object under interaction, our objective is to construct a PhysTwin model that captures the geometry, appearance, and physical parameters of the object over time. At each time frame t, we denote the RGBD observations from the i-th camera as $\mathbf{O}_{t,i}$, where $\mathbf{O} = (\mathbf{I}, \mathbf{D})$ represents the RGB image $\mathbf{I}$ and depth map $\mathbf{D}$.

The goal of our optimization problem is to minimize the discrepancy between the predicted observation $\hat{\mathbf{O}}_{t,i}$ and the actual observation $\mathbf{O}_{t,i}$. The predicted observation is derived by projecting and rendering the predicted state $\hat{\mathbf{X}}_t$ onto images through a function g_θ, where θ encodes the appearance

[1] In practice, we use the first-frame object state as the canonical state.

of the objects represented by Gaussian splats. The 3D state $\hat{\mathbf{X}}_t$ evolves over time according to the Spring-Mass model, which captures the deformable object's dynamics and updates the state using the explicit Euler integration method. The optimization problem is formulated as:

$$\begin{aligned} &\min_{\alpha,\mathcal{G}_0,\theta} \sum_{t,i} C(\hat{\mathbf{O}}_{t,i}, \mathbf{O}_{t,i}) \\ &\text{s.t.} \quad \hat{\mathbf{O}}_{t,i} = g_\theta(\hat{\mathbf{X}}_t, i), \qquad \hat{\mathbf{X}}_{t+1} = f_{\alpha,\mathcal{G}}(\hat{\mathbf{X}}_t, a_t), \end{aligned} \tag{2}$$

where $\alpha, \mathcal{G}_0, \theta$ captures the physics, geometry, topology and appearance parameters (Sec. 3); the cost function quantifies the difference between the predicted observation $\hat{\mathbf{O}}_{t,i}$ and the actual observation $\mathbf{O}_{t,i}$. This cost function is decomposed into three components: $C = C_{\text{geometry}} + C_{\text{motion}} + C_{\text{render}}$, each capturing the discrepancy between the inferred system states and the corresponding observations from 3D geometry, 3D motion tracking, and 2D color, respectively (we defer the details of each cost component to Sec. 4.2.1 and Sec. 4.2.2). The function g_θ is the observation model, describing the projection from the predicted state to the image plane and rendering image-space sensory observation from the i-th camera. The $f_{\alpha,\mathcal{G}}$ models the dynamic evolution of the object's state under the Spring-Mass model (Sec. 3).

4.2. PhysTwin Framework

Given the complexity of the overall optimization defined in Eq. 2, our PhysTwin framework decomposes it into two stages. The first stage focuses on optimizing the geometry and physical parameters, while the second stage is dedicated to optimizing the appearance-related parameters.

4.2.1. Physics and Geometry Optimization

As outlined in our optimization formulation in Sec. 4.1, the objective is to minimize the discrepancy between the predicted observation $\hat{\mathbf{O}}_{t,i}$ and the actual observation $\mathbf{O}_{t,i}$. First, we convert the depth observations $\mathbf{D}_t$ at each time frame t into the observed partial 3D point cloud $\mathbf{X}_t$. In the first stage, we consider the following formulation for the optimization:

$$\begin{aligned} &\min_{\alpha,\mathcal{G}_0} \sum_t \Big(C_{\text{geometry}}(\hat{\mathbf{X}}_t, \mathbf{X}_t) + C_{\text{motion}}(\hat{\mathbf{X}}_t, \mathbf{X}_t) \Big) \\ &\text{s.t.} \quad \hat{\mathbf{X}}_{t+1} = f_{\alpha,\mathcal{G}_0}(\hat{\mathbf{X}}_t, a_t), \end{aligned} \tag{3}$$

where the C_{geometry} function quantifies the single-direction Chamfer distance between the partially observed point cloud $\mathbf{X}_t$ and the inferred state $\hat{\mathbf{X}}_t$, and C_{motion} quantifies the tracking error between the predicted point $\hat{\mathbf{x}}_i^t$ and its corresponding observed tracking $\mathbf{x}_i^t$. The observed tracking is obtained using the vision foundation model CoTracker3 [23], followed by lifting the result to 3D via depth map unprojection.

There are three main challenges in the first-stage optimization: (1) partial observations from sparse viewpoints; (2) joint optimization of both the discrete topology and physical parameters; and (3) discontinuities in the dynamic model, along with the long time horizon and dense parameter space, which make continuous optimization difficult. To address these challenges, we handle the geometry and other parameters separately. Specifically, we first leverage generative shape initialization to obtain the full geometry, then employ our two-stage sparse-to-dense optimization to refine the remaining parameters.

Generative Shape Prior. Due to partial observations, recovering the full geometry is challenging. We leverage a shape prior from the image-to-3D generative model TRELLIS [62] to generate a complete mesh conditioned on a single RGB observation of the masked object. To improve mesh quality, the input to TRELLIS is first enhanced using a super-resolution model [48] that upscales the segmented foreground (obtained via Grounded-SAM2 [46]). While the resulting mesh corresponds reasonably well with the camera observation, we can still observe inconsistencies in scale, pose, and deformation.

To address this, we design a registration module that uses 2D matching for scale estimation, rigid registration, and non-rigid deformation. A coarse-to-fine strategy first estimates initial rotation via 2D correspondences matched using SuperGlue [50], followed by refinement with the Perspective-n-Point (PnP) [25] algorithm. We resolve scale and translation ambiguities by optimizing the distances between matched points in the camera coordinate system. After applying these transformations, the objects are aligned in pose, with some deformations handled by as-rigid-as-possible registration [53]. Finally, ray-casting alignment ensures that observed points match the deformed mesh without occlusions.

These steps yield a shape prior aligned with the first-frame observations, which serves as a crucial initialization for the inverse physics and appearance optimization stages.

Sparse-to-Dense Optimization. The Spring-Mass model consists of both the topological structure (i.e., the connectivity of the springs) and the physical parameters defined on the springs. As mentioned in Sec. 3, we also include control parameters to connect springs between control points and object points, defined by a radius and a maximum number of neighbors. Similarly, for topology optimization, we employ a heuristic approach that connects nearest-neighbor points, also parameterized by a connection radius and a maximum number of neighbors, thereby controlling the density of the springs. To extract control points from video data, we utilize Grounded-SAM2 [46] to segment the hand mask and CoTracker3 [23] to track hand movements. After lifting the points to 3D, we apply farthest-point sampling to obtain the final set of control points.

All the aforementioned components constitute the parameter space we aim to optimize. The two main challenges are: (1) some parameters are non-differentiable (e.g., the radius

and maximum number of neighbors); and (2) to represent a wide range of objects, we model dense spring stiffness, leading to a parameter space with tens of thousands of springs.

To address these challenges, we introduce a hierarchical sparse-to-dense optimization strategy. Initially, we employ zero-order, sampling-based optimization to estimate the parameters, which naturally circumvents the issue of differentiability. However, zero-order optimization becomes inefficient when the parameter space is too large. Therefore, in the first stage, we assume homogeneous stiffness, allowing the topology and other physical parameters to achieve a good initialization. In the second stage, we further refine the parameters using first-order gradient descent, leveraging our custom-built differentiable spring-mass simulator. This stage simultaneously optimizes the dense spring stiffness and collision parameters.

Beyond the optimization strategy, we incorporate additional supervision by utilizing tracking priors from vision foundation models. We lift the 2D tracking prediction into 3D to obtain pseudo-ground-truth tracking data for the 3D points, which forms a crucial component of our cost function as mentioned in Eq. (3).

By integrating our optimization strategy with a cost function that leverages additional tracking priors, our PhysTwin framework can effectively and efficiently model the dynamics of diverse interactable objects from videos.

4.2.2. Appearance Optimization

For the second-stage appearance optimization, to model object appearance, we construct a set of static 3D Gaussian kernels parameterized by θ, with each Gaussian defined by a 3D center position μ, a rotation matrix represented by a quaternion $q \in \mathbf{SO}(3)$, a scaling matrix represented by a 3D vector s, an opacity value α, and color coefficients c. We optimize θ here via

$$\min_{\theta} \sum_{t,i} C_{\text{render}}(\hat{\mathbf{I}}_{i,t}, \mathbf{I}_{i,t}) \text{ s.t. } \hat{\mathbf{I}}_{i,t} = g_{\theta}(\hat{\mathbf{X}}_t, i), \quad (4)$$

where $\hat{\mathbf{X}}_t$ is the optimized system states at time t, i is the camera index, and $\mathbf{I}_{i,t}$, $\hat{\mathbf{I}}_{i,t}$ are the ground truth image and rendered image from camera view i at time t, respectively. C_{render} computes the $\mathcal{L}_1$ loss with a D-SSIM term between the rendering and ground truth image. For simplicity, we set $t = 0$ to optimize appearance only at the first frame. We restrict the Gaussian shape to be isotropic to prevent spiky artifacts during deformation.

To ensure realistic rendering under deformation, we need to dynamically adjust each Gaussian at each timestep t based on the transition between states $\hat{\mathbf{X}}_t$ and $\hat{\mathbf{X}}_{t+1}$. To achieve this, we adopt a Gaussian updating algorithm using Linear Blend Skinning (LBS) [20, 54, 69], which interpolates the motions of 3D Gaussians using the motions of neighboring mass nodes. Please refer to the supplementary for details.

4.3. Capabilities of PhysTwin

Our constructed PhysTwin supports real-time simulation of deformable objects under various motions while maintaining realistic appearance. This real-time, photorealistic simulation enables interactive exploration of object dynamics.

By introducing control points and dynamically connecting them to object points via springs, our system can simulate diverse motion patterns and interactions. These capabilities make PhysTwin a powerful representation for real-time interactive simulation and model-based robotic motion planning, which are further described in Sec. 5.3.

5. Experiments

In this section, we evaluate the performance of our PhysTwin framework across three distinct tasks involving different types of objects. Our primary objective is to address the following three questions: (1) How accurately does our framework reconstruct and resimulate deformable objects and predict their future states? (2) How well does the constructed PhysTwin generalize to unseen interactions? (3) What is the utility of PhysTwin in downstream tasks?

5.1. Experiment Settings

Dataset. We collect a dataset of RGBD videos capturing human interactions with various deformable objects with different physical properties, such as ropes, stuffed animals, cloth, and delivery packages. Three RealSense-D455 RGBD cameras are used to record the interactions. Each video is 1 to 10 seconds long and captures different interactions, including quick lifting, stretching, pushing, and squeezing with one or both hands. We collect 22 scenarios encompassing various object types, interaction types, and hand configurations. For each scenario, the RGBD videos are split into a training set and a test set following a 7:3 ratio, where only the training set is used to construct PhysTwin. We manually annotate 9 ground-truth tracking points for each video to evaluate tracking performance with the semi-auto tool introduced in [7].

Tasks. To assess the effectiveness of our PhysTwin framework and the quality of our constructed PhysTwin, we formulate three tasks: (1) Reconstruction & Resimulation; (2) Future Prediction; and (3) Generalization to Unseen Actions.

For the Reconstruction & Resimulation task, the objective is to construct PhysTwin such that it can accurately reconstruct and resimulate the motion of deformable objects given the actions represented by the control point positions.

For the Future Prediction task, we aim to assess whether PhysTwin can perform well on unseen future frames during its construction. For the Generalization to Unseen Interactions task, the goal is to assess whether PhysTwin can adapt to different interactions. To evaluate this, we construct a generalization dataset consisting of interaction pairs performed

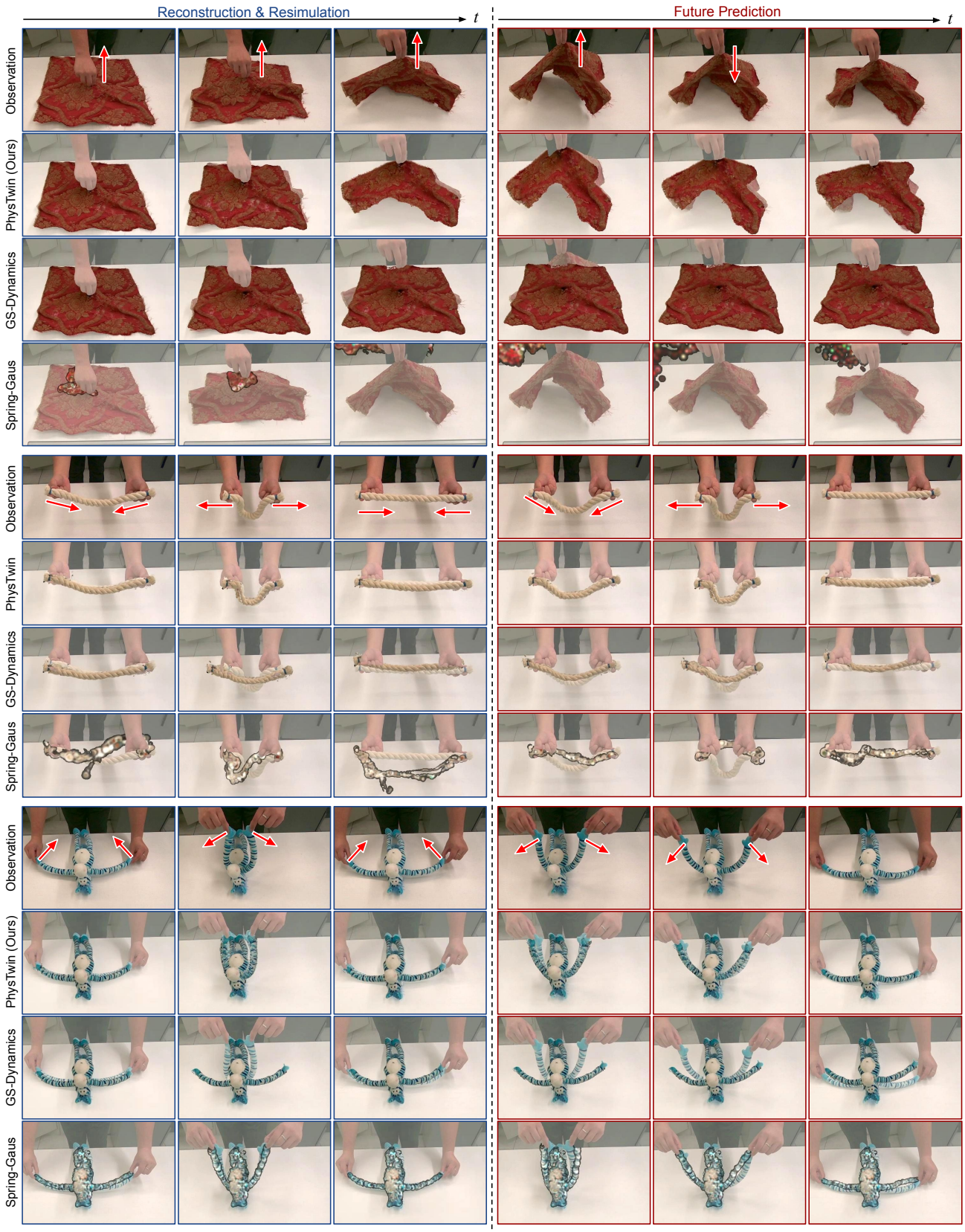


Figure 3. **Qualitative Results on Reconstruction & Resimulation and Future Prediction.** We visualize the rendering results of different methods on two tasks. For the reconstruction & resimulation task, our method achieves a better match with the observations. For the future prediction task, our method accurately predicts the future state of the objects. In contrast, the baselines fail in most cases: GS-Dynamics [69] tends to remain static, while Spring-Gauss [72] frequently causes the physical model to crash.

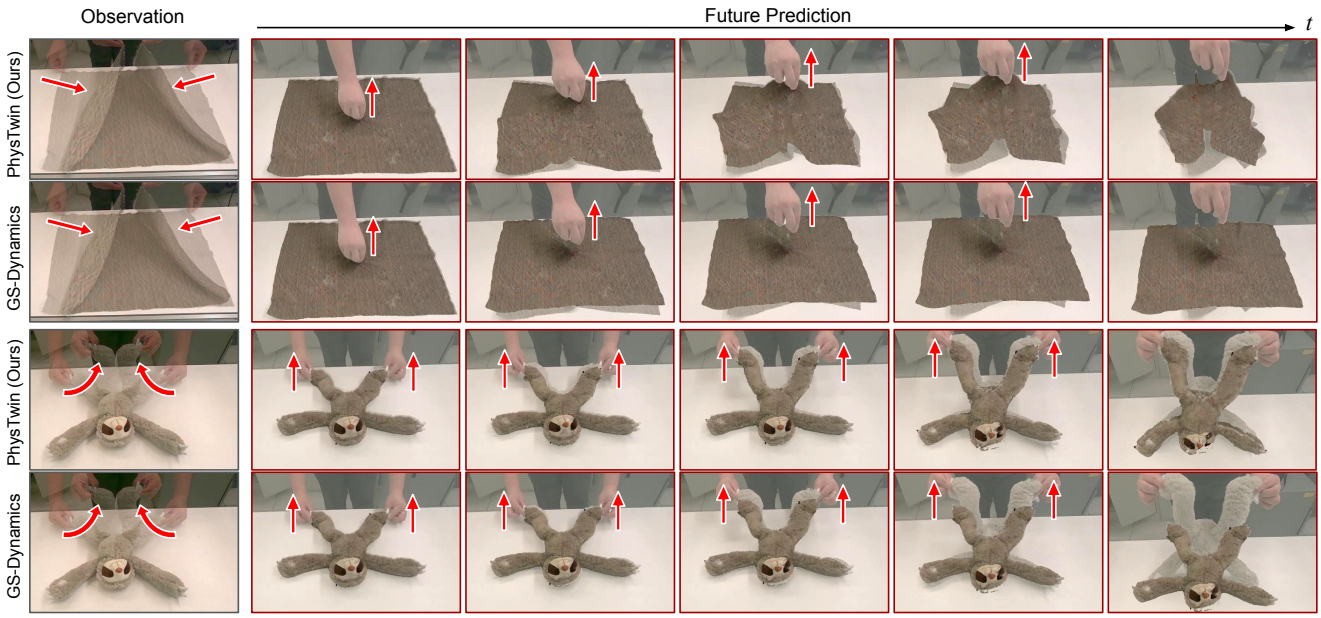


Figure 4. **Qualitative Results on Generalization to Unseen Interactions.** We visualize the simulation of a deformable object under unseen interactions using our method and GS-Dynamics [69]. The leftmost image shows the interaction used to train the dynamics models, while the images on the right demonstrate their generalization to unseen interactions. Our PhysTwin significantly outperforms prior work.

Table 1. **Quantitative Results on Reconstruction & Resimulation and Future Prediction.** We compare the performance of our method with two prior work, GS-Dynamics [69] and Spring-Gaus [72], on two tasks: reconstruction & resimulation and future prediction. Our PhysTwin framework consistently outperforms the baselines across all metrics.

Task	Reconstruction & Resimulation						Future Prediction					
Method	CD ↓	Track Error ↓	IoU % ↑	PSNR ↑	SSIM ↑	LPIPS ↓	CD ↓	Track Error ↓	IoU % ↑	PSNR ↑	SSIM ↑	LPIPS ↓
Spring-Gaus [72]	0.041	0.050	57.6	23.445	0.928	0.102	0.062	0.094	46.4	22.488	0.924	0.113
GS-Dynamics [69]	0.014	0.022	72.1	26.260	0.940	0.052	0.041	0.070	49.8	22.540	0.924	0.097
PhysTwin (**Ours**)	**0.005**	**0.009**	**84.4**	**28.214**	**0.945**	**0.034**	**0.012**	**0.022**	**72.5**	**25.617**	**0.941**	**0.055**

Table 2. **Quantitative Results on Generalization to Unseen Interactions.** We compare our method with GS-Dynamics [69] on generalization to unseen interactions. Both methods are trained on the same video with a specific interaction and tested on unseen interactions. Our method achieves significantly better results.

Method	CD ↓	Track Error ↓	IoU % ↑	PSNR ↑	SSIM ↑	LPIPS ↓
GS-Dynamics [69]	0.029	0.038	63.4	25.053	0.934	0.067
PhysTwin (**Ours**)	**0.013**	**0.018**	**72.18**	**26.199**	**0.938**	**0.047**

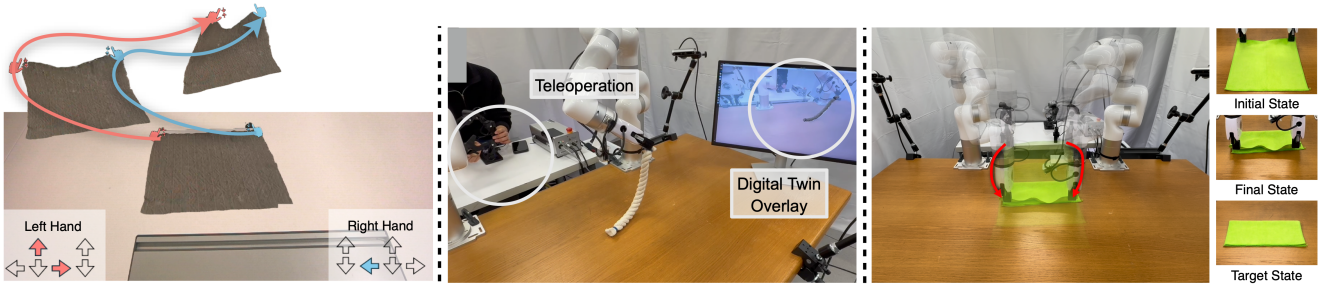
on the same object but with varying motions, including differences in hand configuration and interaction type.

Baselines. To the best of our knowledge, there is currently no existing work that demonstrates good performance across all three tasks. Therefore, we select two main research directions as baselines and further augment them to match the tasks in our setting (full details in the supplementary).

The first baseline we consider is a physics-based simulation method for identifying the material properties of deformable objects, Spring-Gaus [72]. Their work has demonstrated strong capabilities in reconstruction, resimulation, and future prediction in its original setting. However, their framework does not support external control inputs, so we augment it with additional control capabilities.

The second baseline is a learning-based simulation approach, GS-Dynamics [69], which employs a GNN-based neural dynamics model to learn system dynamics directly from partial observations. In their original setting, video preprocessing with Dyn3DGS [34] is required to obtain tracking information. For a fairer comparison, we strengthened it by using our 3D-lifting tracker based on CoTracker3 [23], which provides more efficient and accurate supervision for training the neural dynamics model used by GS-Dynamics.

Evaluation. To better understand whether our prediction matches the observations, we evaluate predictions in both 3D and 2D. For the 3D evaluation, we use the single-direction Chamfer Distance (partial ground truth with our full-state prediction) and the tracking error (based on our manually annotated ground-truth tracking points). For the 2D evaluation, we assess image quality using PSNR, SSIM, and LPIPS [70], and silhouette alignment using IoU. We perform 2D evaluation only at the center viewpoint due to optimal visibility of objects, with metrics averaged across all frames and scenarios. Specially, for the Spring-Gaus [72] baseline, its optimization process is unstable due to inaccurate physics modeling. Therefore, we report the above metrics only for its successful cases.



(a) Real-Time Interactive Simulation using Keyboard (b) Real-Time Interactive Simulation using Teleoperation (c) Model-Based Robot Planning

Figure 5. **Applications of our PhysTwin.** Our constructed PhysTwin supports a variety of tasks, including real-time interactive simulation, which can accept input from either a keyboard or a robot teleoperation setup. Meanwhile, PhysTwin also enables model-based robot planning to accomplish tasks such as lifting a rope into some specific configuration.

5.2. Results

To assess the performance of our framework and the quality of our constructed PhysTwin, we compare with two augmented baselines across three task settings. Our quantitative analysis reveals that the PhysTwin framework consistently outperforms the baselines across various tasks.

Reconstruction & Resimulation. The quantitative results in Tab. 1 Reconstruction & Resimulation column demonstrate the superior performance of our PhysTwin method over baselines. Our approach significantly improves all evaluated metrics, including Chamfer Distance, tracking error, and 2D IoU, confirming that our reconstruction and resimulation align more closely with the original observations. This highlights the effectiveness of our model in learning a more accurate dynamics model under sparse observations. Additionally, rendering metrics show that our method produces more realistic 2D images, benefiting from the Gaussian blending strategy and enhanced dynamic modeling. Fig. 3 further provides qualitative visualizations across different objects, illustrating precise alignment with original observations. Notably, our physics-based representation inherently improves point tracking. After physics-constrained optimization, our tracking surpasses the original CoTracker3 [23] predictions used for training, achieving better alignment after global optimization (See supplement for more details).

Future Prediction. Table 1, in the Future Prediction column, demonstrates that our method achieves superior performance in predicting unseen frames, excelling in both dynamics alignment and rendering quality. Fig. 3 further provides qualitative results, illustrating the accuracy of our predictions on unseen frames.

Generalization to Unseen Interactions. We also evaluate the generalization performance to unseen interactions. Our dataset includes transfers from one interaction (e.g., single lift) to significantly different interactions (e.g., double stretch). We directly use our constructed PhysTwin and leverage our registration pipeline to align it with the first frame of the target case. Fig. 4 shows that our method closely matches the ground truth observations in terms of dynamics. Quantitative results further demonstrate the robustness of our method across different actions. In contrast, the neural dynamics model struggles to adapt to environmental changes and diverse interactions as effectively as our approach. Moreover, in unseen interaction scenarios, our method achieves performance comparable to that on the future prediction task, highlighting the robustness and generalization capability of our constructed PhysTwin.

5.3. Application

The efficient forward simulation capabilities of our Spring-Mass simulator, implemented using Warp [37], enable a variety of downstream applications. Fig. 5 showcases key applications enabled by our PhysTwin: (1) Interactive Simulation: Users can interact with objects in real time using keyboard controls, either with one or both hands. The system also supports real-time simulation of an object's future state during human teleoperation with robotic arms. This feature serves as a valuable tool for predicting object dynamics during manipulation. (2) Model-Based Robotic Planning: Owing to the high fidelity of our constructed PhysTwin, it can be used as a dynamic model in planning pipelines. By integrating it with model-based planning techniques, we can generate effective motion plans for robots to complete a variety of tasks.

6. Conclusion

We introduced PhysTwin, a novel framework for constructing physical digital twins from sparse videos, enabling effective reconstruction and resimulation of deformable objects. Our approach excels in predicting future states and simulating object interactions that generalize to unseen actions. We showed the superior performance of our method across various object types, control configurations, and task settings, significantly outperforming prior work. PhysTwin enables various downstream tasks that demand high-speed simulation and accurate future prediction. Moreover, our approach provides valuable insights for robotic manipulation. By bridging perception and physics-based simulation, PhysTwin serves as a crucial tool for guiding robot interactions, making real-world deployment more efficient and reliable.

Acknowledgement

This work is partially supported by the Toyota Research Institute (TRI), the Sony Group Corporation, Google, Dalus AI, the DARPA TIAMAT program (HR0011-24-9-0430), the Intel AI SRS gift, Amazon-Illinois AICE grant, Meta Research Grant, IBM IIDAI Grant, and NSF Awards #2331878, #2340254, #2312102, #2414227, and #2404385. We greatly appreciate the NCSA for providing computing resources. This article solely reflects the opinions and conclusions of its authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsors.

References

[1] Benjamin Attal, Jia-Bin Huang, Christian Richardt, Michael Zollhoefer, Johannes Kopf, Matthew O'Toole, and Changil Kim. Hyperreel: High-fidelity 6-dof video with ray-conditioned sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16610–16620, 2023. 2

[2] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 130–141, 2023. 1, 2

[3] Hsiao-yu Chen, Edith Tretschk, Tuur Stuyck, Petr Kadlecek, Ladislav Kavan, Etienne Vouga, and Christoph Lassner. Virtual elastic objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15827–15837, 2022. 2

[4] 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. *arXiv preprint arXiv:2205.01089*, 2022. 1, 2

[5] Mengyu Chu, Lingjie Liu, Quan Zheng, Erik Franz, Hans-Peter Seidel, Christian Theobalt, and Rhaleb Zayer. Physics informed neural fields for smoke reconstruction with sparse data. *ACM Transactions on Graphics*, 41(4):119:1–119:14, 2022. 1, 2

[6] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996. 2

[7] Carl Doersch, Yi Yang, Mel Vecerik, Dilara Gokay, Ankush Gupta, Yusuf Aytar, Joao Carreira, and Andrew 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. 5

[8] Danny Driess, Zhiao Huang, Yunzhu Li, Russ Tedrake, and Marc Toussaint. Learning multi-object dynamics with compositional neural radiance fields. In *Conference on robot learning*, pages 1755–1768. PMLR, 2023. 1, 2

[9] Tao Du, Kui Wu, Pingchuan Ma, Sebastien Wah, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. Diffpd: Differentiable projective dynamics. *ACM Transactions on Graphics (ToG)*, 41(2):1–21, 2021. 1, 2

[10] Yuanxing Duan, Fangyin Wei, Qiyu Dai, Yuhang He, Wenzheng Chen, and Baoquan Chen. 4d-rotor gaussian splatting: towards efficient novel view synthesis for dynamic scenes. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 1, 2

[11] Ben Evans, Abitha Thankaraj, and Lerrel Pinto. Context is everything: Implicit identification for dynamics adaptation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2642–2648. IEEE, 2022. 1, 2

[12] Yutao Feng, Yintong Shang, Xuan Li, Tianjia Shao, Chenfanfu Jiang, and Yin Yang. Pie-nerf: Physics-based interactive elastodynamics with nerf. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4450–4461, 2024. 1, 2

[13] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12479–12488, 2023. 1, 2

[14] Hang Gao, Ruilong Li, Shubham Tulsiani, Bryan Russell, and Angjoo Kanazawa. Monocular dynamic view synthesis: A reality check. *Advances in Neural Information Processing Systems*, 35:33768–33780, 2022. 1, 2

[15] Moritz Geilinger, David Hahn, Jonas Zehnder, Moritz Bächer, Bernhard Thomaszewski, and Stelian Coros. Add: Analytically differentiable dynamics for multi-body systems with frictional contact. *ACM Transactions on Graphics (TOG)*, 39 (6):1–15, 2020. 2

[16] Shanyan Guan, Huayu Deng, Yunbo Wang, and Xiaokang Yang. Neurofluid: Fluid dynamics grounding with particle-driven neural radiance fields. In *International conference on machine learning*, pages 7919–7929. PMLR, 2022. 2

[17] Xiang Guo, Jiadai Sun, Yuchao Dai, Guanying Chen, Xiaoqing Ye, Xiao Tan, Errui Ding, Yumeng Zhang, and Jingdong Wang. Forward flow for novel view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16022–16033, 2023. 1, 2

[18] Nikolaus Hansen. The cma evolution strategy: a comparing review. *Towards a new evolutionary computation: Advances in the estimation of distribution algorithms*, pages 75–102, 2006. 2

[19] Eric Heiden, Miles Macklin, Yashraj Narang, Dieter Fox, Animesh Garg, and Fabio Ramos. Disect: A differentiable simulation engine for autonomous robotic cutting. *arXiv preprint arXiv:2105.12244*, 2021. 2

[20] Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4220–4230, 2024. 1, 2, 5, 12

[21] Krishna Murthy Jatavallabhula, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jérôme Parent-Lévesque, Kevin Xie, Kenny Erleben, et al. gradsim: Differentiable simulation for system identification and visuomotor control. *arXiv preprint arXiv:2104.02646*, 2021. 2

[22] Ying Jiang, Chang Yu, Tianyi Xie, Xuan Li, Yutao Feng, Huamin Wang, Minchen Li, Henry Lau, Feng Gao, Yin Yang, et al. Vr-gs: A physical dynamics-aware interactive gaussian splatting system in virtual reality. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–1, 2024. 2

[23] Nikita Karaev, Iurii Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker3: Simpler and better point tracking by pseudo-labelling real videos. *arXiv preprint arXiv:2410.11831*, 2024. 2, 4, 7, 8, 12, 16

[24] Agelos Kratimenos, Jiahui Lei, and Kostas Daniilidis. Dynmf: Neural motion factorization for real-time dynamic view synthesis with 3d gaussian splatting. In *European Conference on Computer Vision*, pages 252–269. Springer, 2024. 1, 2

[25] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Ep n p: An accurate o (n) solution to the p n p problem. *International journal of computer vision*, 81:155–166, 2009. 4

[26] Hao Li, Robert W Sumner, and Mark Pauly. Global correspondence optimization for non-rigid registration of depth scans. In *Computer graphics forum*, pages 1421–1430. Wiley Online Library, 2008. 2

[27] Xuan Li, Yi-Ling Qiao, Peter Yichen Chen, Krishna Murthy Jatavallabhula, Ming Lin, Chenfanfu Jiang, and Chuang Gan. Pac-nerf: Physics augmented continuum neural radiance fields for geometry-agnostic system identification. *arXiv preprint arXiv:2303.05512*, 2023. 1, 2

[28] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. *arXiv preprint arXiv:1810.01566*, 2018. 1, 2

[29] Yunzhu Li, Shuang Li, Vincent Sitzmann, Pulkit Agrawal, and Antonio Torralba. 3d neural scene representations for visuomotor control. In *Conference on Robot Learning*, pages 112–123. PMLR, 2022. 1, 2

[30] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6498–6508, 2021. 2

[31] Zhengqi Li, Qianqian Wang, Forrester Cole, Richard Tucker, and Noah Snavely. Dynibar: Neural dynamic image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4273–4284, 2023. 1, 2

[32] Xingyu Lin, Yufei Wang, Zixuan Huang, and David Held. Learning visible connectivity dynamics for cloth smoothing. In *Conference on Robot Learning*, pages 256–266. PMLR, 2022. 1, 2

[33] Youtian Lin, Zuozhuo Dai, Siyu Zhu, and Yao Yao. Gaussian-flow: 4d reconstruction with dynamic 3d gaussian particle. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21136–21145, 2024. 1, 2

[34] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *2024 International Conference on 3D Vision (3DV)*, pages 800–809. IEEE, 2024. 1, 2, 7

[35] 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. *arXiv preprint arXiv:2205.05678*, 2022. 2

[36] Pingchuan Ma, Peter Yichen Chen, Bolei Deng, Joshua B Tenenbaum, Tao Du, Chuang Gan, and Wojciech Matusik. Learning neural constitutive laws from motion observations for generalizable pde dynamics. In *International Conference on Machine Learning*, pages 23279–23300. PMLR, 2023. 1, 2

[37] Miles Macklin. Warp: A high-performance python framework for gpu simulation and graphics. https://github.com/nvidia/warp, 2022. NVIDIA GPU Technology Conference (GTC). 8

[38] Richard A Newcombe, Dieter Fox, and Steven M Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 343–352, 2015. 2

[39] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5865–5874, 2021. 1, 2

[40] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *arXiv preprint arXiv:2106.13228*, 2021. 2

[41] Yicong Peng, Yichao Yan, Shenqi Liu, Yuhao Cheng, Shanyan Guan, Bowen Pan, Guangtao Zhai, and Xiaokang Yang. Cagenerf: Cage-based neural radiance fields for generalized 3d deformation and animation. In *Thirty-Sixth Conference on Neural Information Processing Systems*, 2022. 1, 2

[42] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International conference on learning representations*, 2020. 1, 2

[43] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10318–10327, 2021. 1, 2

[44] Yi-Ling Qiao, Junbang Liang, Vladlen Koltun, and Ming C. Lin. Differentiable simulation of soft multi-body systems. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021. 1, 2

[45] Yi-Ling Qiao, Alexander Gao, and Ming Lin. Neuphysics: Editable neural geometry and physics from monocular videos. *Advances in Neural Information Processing Systems*, 35: 12841–12854, 2022. 2

[46] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, et al. Grounded sam: Assembling open-world models for diverse visual tasks. *arXiv preprint arXiv:2401.14159*, 2024. 2, 4

[47] Junior Rojas, Eftychios Sifakis, and Ladislav Kavan. Differentiable implicit soft-body physics. *arXiv preprint arXiv:2102.05791*, 2021. 2

[48] 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*, pages 10684–10695, 2022. 2, 4

[49] 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*, pages 8459–8468. PMLR, 2020. 1, 2

[50] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020. 4, 12

[51] Haochen Shi, Huazhe Xu, Samuel Clarke, Yunzhu Li, and Jiajun Wu. Robocook: Long-horizon elasto-plastic object manipulation with diverse tools. *arXiv preprint arXiv:2306.14447*, 2023. 1, 2

[52] Haochen Shi, Huazhe Xu, Zhiao Huang, Yunzhu Li, and Jiajun Wu. Robocraft: Learning to see, simulate, and shape elasto-plastic objects in 3d with graph networks. *The International Journal of Robotics Research*, 43(4):533–549, 2024. 1, 2

[53] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, pages 109–116. Citeseer, 2007. 4

[54] Robert W Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. In *SIGGRAPH*, pages 80–es. 2007. 5, 12

[55] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12959–12970, 2021. 1, 2

[56] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2021. 1, 2

[57] Bin Wang, Longhua Wu, KangKang Yin, Uri M Ascher, Libin Liu, and Hui Huang. Deformation capture and modeling of soft objects. *ACM Trans. Graph.*, 34(4):94–1, 2015. 2

[58] Chaoyang Wang, Lachlan Ewen MacDonald, Laszlo A Jeni, and Simon Lucey. Flow supervision for deformable nerf. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21128–21137, 2023. 1, 2

[59] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20310–20320, 2024. 1, 2

[60] Yilin Wu, Wilson Yan, Thanard Kurutach, Lerrel Pinto, and Pieter Abbeel. Learning to manipulate deformable objects without demonstrations. *arXiv preprint arXiv:1910.13439*, 2019. 1, 2

[61] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9421–9431, 2021. 1, 2

[62] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. *arXiv preprint arXiv:2412.01506*, 2024. 2, 4, 12

[63] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4389–4398, 2024. 1, 2

[64] Zhenjia Xu, Jiajun Wu, Andy Zeng, Joshua B Tenenbaum, and Shuran Song. Densephysnet: Learning dense physical object representations via multi-step dynamic interactions. *arXiv preprint arXiv:1906.03853*, 2019. 1, 2

[65] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. *arXiv preprint arXiv:2310.10642*, 2023. 1, 2

[66] 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*, pages 20331–20341, 2024. 1, 2

[67] Heng Yu, Joel Julin, Zoltan A Milacski, Koichiro Niinuma, and Laszlo A Jeni. Dylin: Making light field networks dynamic. *arXiv preprint arXiv:2303.14243*, 2023. 2

[68] Heng Yu, Joel Julin, Zoltán Á Milacski, Koichiro Niinuma, and László A Jeni. Cogs: Controllable gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21624–21633, 2024. 1, 2

[69] Mingtong Zhang, Kaifeng Zhang, and Yunzhu Li. Dynamic 3d gaussian tracking for graph-based neural dynamics modeling. *arXiv preprint arXiv:2410.18912*, 2024. 1, 2, 5, 6, 7, 12, 14

[70] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 7

[71] Tianyuan Zhang, Hong-Xing Yu, Rundi Wu, Brandon Y Feng, Changxi Zheng, Noah Snavely, Jiajun Wu, and William T Freeman. Physdreamer: Physics-based interaction with 3d objects via video generation. In *European Conference on Computer Vision*, pages 388–406. Springer, 2024. 1, 2

[72] Licheng Zhong, Hong-Xing Yu, Jiajun Wu, and Yunzhu Li. Reconstruction and simulation of elastic objects with spring-mass 3d gaussians. In *European Conference on Computer Vision*, pages 407–423. Springer, 2024. 1, 2, 6, 7, 14

Supplement Index

In the supplement, we provide additional details of our PhysTwin framework, more qualitative results across different tasks, and further analysis of our methods. All the videos showcasing our results on various instances, interactions, and tasks are available on our website.

A. Additional Details for the Shape Prior

As mentioned in the main paper, we leverage TRELLIS [62] to generate the full mesh from a single RGB observation. However, the potential non-rigid registration presents a non-trivial challenge.

To address these issues, we design a registration module that leverages 2D matching to handle scale estimation, rigid registration, and non-rigid deformation. First, to estimate the initial rotation, we adopt a coarse-to-fine strategy. We use uniformly distributed virtual cameras placed on a sphere surrounding the object to render images and match 2D correspondences using SuperGlue [50]. Based on the number of matches, we select the view with the maximum number of correspondences, providing a rough rotation estimate. We then apply the Perspective-n-Point (PNP) algorithm to refine the 3D matched points on the generated mesh and the corresponding 2D pixels in the observation, estimating the precise rotation matrix.

After estimating the rotation, translation and scale ambiguities may still exist. To resolve these, we optimize the distances between matched point pairs to solve for scale and translation. This is simplified in the camera coordinate system, as after PNP, the matched points in the generated mesh and the corresponding points in the real observation point cloud lie along the same line connecting the origin. Therefore, the scale and translation optimization can be reduced to optimizing only the scale. Once these transformations are applied, the two objects should be in similar poses, with some parts undergoing non-rigid deformations. To handle such deformations, we use an as-rigid-as-possible registration to deform the mesh into a non-rigid pose matching the real observation. Finally, we perform ray-casting alignment, shooting rays from the camera to ensure that the observed points align with the deformed mesh and are neither occluded nor occlude the mesh.

B. Additional Details for 3D Gaussian Update

Given the previous state $\hat{\mathbf{X}}_t$ and the predicted state $\hat{\mathbf{X}}_{t+1}$, we first solve for the 6-DoF transformation of each mass node $\hat{\mu}_i^t \in \hat{\mathbf{X}}_t$. For 3D translations, we obtain them from the predicted node translations T_i^t. For 3D rotations, for each vertex $\hat{\mu}_i^t$, we estimate a rigid local rotation R_i^t based on motions of its neighbors $\mathcal{N}(i)$ from time t to $t+1$:

$$R_i^t = \arg\min_{R \in SO(3)} \sum_{j \in \mathcal{N}(i)} \|R(\hat{\mu}_j^t - \hat{\mu}_i^t) - (\hat{\mu}_j^{t+1} - \hat{\mu}_i^{t+1})\|^2. \tag{5}$$

In the next step, we transform Gaussian kernels using Linear Blend Skinning (LBS) [20, 54, 69] by locally interpolating the transformations of their neighboring nodes. Specifically, for the 3D center and rotation of each Gaussian:

$$\mu_j^{t+1} = \sum_{k \in \mathcal{N}(j)} w_{jk}^t (R_k^t (\mu_j^t - \hat{\mu}_k^t) + \hat{\mu}_k^t + T_k^t) \tag{6}$$

$$q_j^{t+1} = (\sum_{k \in \mathcal{N}(j)} w_{jk}^t r_k^t) \otimes q_j^t, \tag{7}$$

where $R_k^t \in \mathbb{R}^{3\times3}$ and $r_k^t \in \mathbb{R}^4$ are the matrix and quaternion forms of the rotation of vertex k; $\otimes$ denotes the quaternion multiply operator; $\mathcal{N}(j)$ represents K-nearest vertices of a Gaussian center μ_j^t; w_{jk}^t is the interpolation weights between a Gaussian μ_j^t and a corresponding vertex $\hat{\mu}_k^t$, which are derived inversely proportional to their 3D distance:

$$w_{jk}^t = \frac{\|\mu_j^t - \hat{\mu}_k\|^{-1}}{\sum_{k \in \mathcal{N}(j)} \|\mu_j^t - \hat{\mu}_k\|^{-1}} \tag{8}$$

to ensure larger weights are assigned to the spatially closer pairs. Finally, with the updated Gaussian parameters, we are able to perform rendering at timestep $t+1$ with the transformed 3D Gaussians.

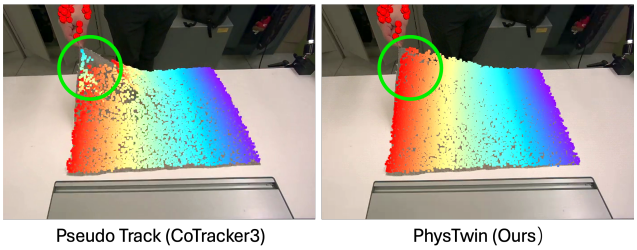


Figure 6. **Visualization of Tracking Results.** We compare the tracking results produced by our PhysTwin with the raw tracking results from CoTracker3 [23]. Our PhysTwin achieves more natural and smoother movement compared to the raw predictions from CoTracker3.

C. Additional Experimental Details

Due to the page limit in the main paper, we provide additional qualitative results on different instances under various interactions, as well as further analysis experiments.

Figure 7. **Additional Qualitative Results on Reconstruction & Resimulation and Future Prediction.**

Table 3. **Ablations of Our Sparse-to-Dense Optimization.** To better understand our optimization process, we conduct ablation experiments comparing results with only zero-order optimization or first-order optimization. The results demonstrate that our sparse-to-dense optimization strategy is effective in obtaining the most accurate physical parameters.

Task	Reconstruction & Resimulation						Future Prediction					
Method	CD ↓	Track Error ↓	IoU % ↑	PSNR ↑	SSIM ↑	LPIPS ↓	CD ↓	Track Error ↓	IoU % ↑	PSNR ↑	SSIM ↑	LPIPS ↓
Zero-order Only	0.007	0.012	80.2	27.409	0.943	0.039	0.014	0.025	69.2	25.008	0.938	0.061
First-order Only	0.008	0.012	82.7	27.913	0.944	0.037	0.019	0.034	65.7	24.572	0.936	0.067
PhysTwin (**Ours**)	**0.005**	**0.009**	**84.4**	**28.214**	**0.945**	**0.034**	**0.012**	**0.022**	**72.5**	**25.617**	**0.941**	**0.055**

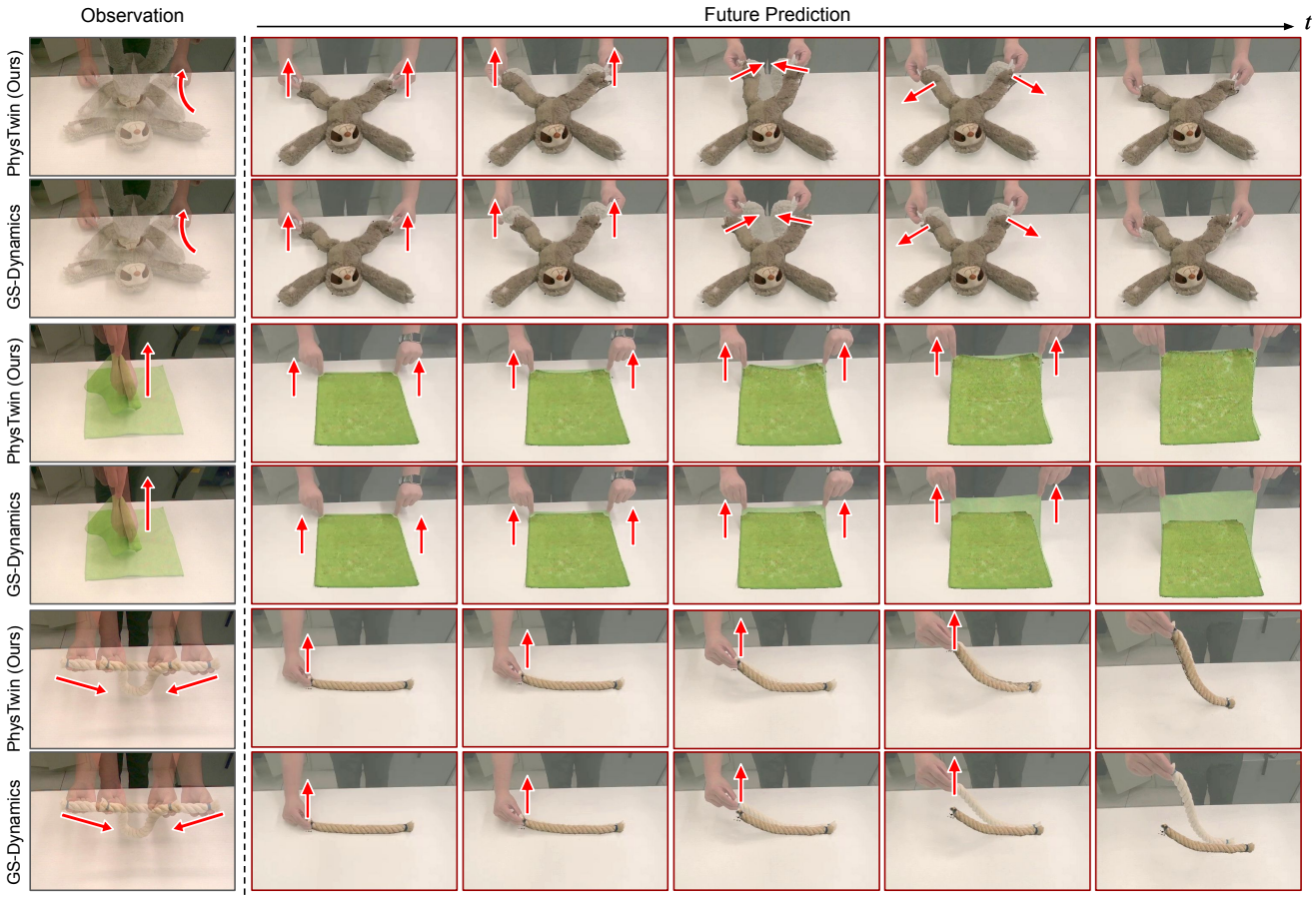


Figure 8. **Additional Qualitative Results on Generalization to Unseen Interactions.**

Baselines. As described in the main paper, we select two prior works for comparison: Spring-Gaus [72] and GS-Dynamics [69].

For Spring-Gaus, while it demonstrates reasonable performance in modeling object-collision videos, its applicability is limited to relatively simple cases where objects primarily deform under gravity, restricting the range of supported object types. To adapt Spring-Gaus [72] to our setting, we extend it by introducing support for control points. Specifically, we add additional springs that connect the control points to their neighboring object points within a predefined distance, enabling direct optimization on our dataset. Furthermore, to ensure compatibility with our sparse-view setup, we incorporate our shape prior as the initialization for their static Gaussian construction. Since their constructed Gaussians lack the ability to generalize to different initial conditions, we evaluate their approach only on the first two tasks: reconstruction & resimulation and future prediction.

For GS-Dynamics, we compare our method with theirs across all three tasks. To enable the GNN-based dynamics model to produce realistic renderings, we augment it with our Gaussian blending strategy, enhancing its ability to generate high-quality images.

Tasks. PhysTwin is constructed solely from the training set of each data point, and its performance is evaluated based on how well it matches the original video within the test set.

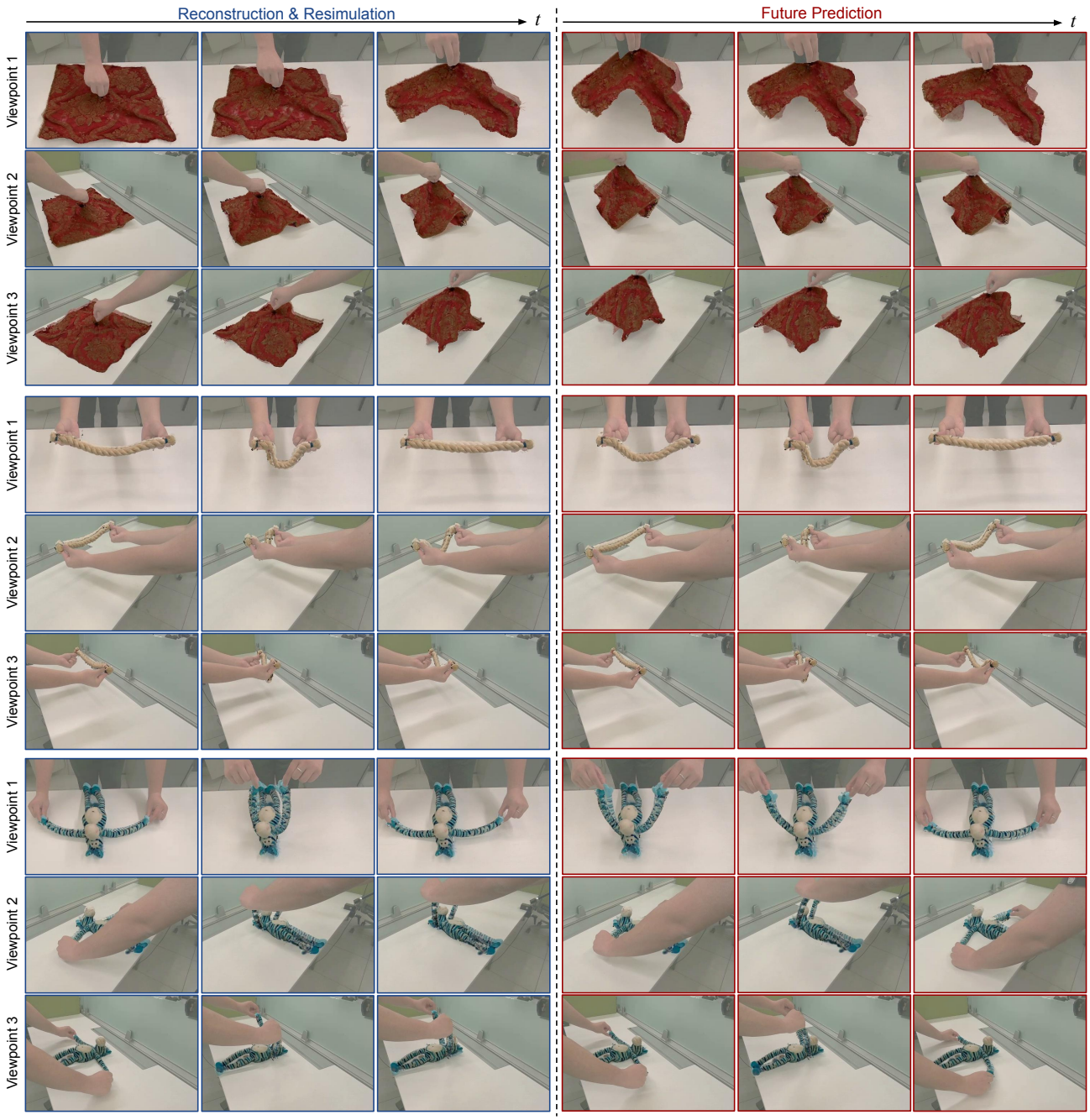


Figure 9. **Qualitative Results on Reconstruction & Resimulation and Future Prediction with different viewpoints.**

For the generalization task, we create a dataset consisting of interaction pairs performed on the same object. For example, we construct PhysTwin for a sloth toy based on a scenario where it is lifted with one hand and then evaluate its performance in a different scenario where its legs are stretched using both hands. The dataset includes 11 such pairs, and since each pair allows for two possible transfer directions (i.e., from one interaction to another or vice versa), this results in a total of 22 generalization experiments. In this task, PhysTwin is still constructed using only the training set of the source interaction but is applied across the entire sequence of the target interaction.

Qualitative Results. We present more qualitative results for different instances across various interactions on our three tasks: reconstruction & resimulation, future prediction (Fig. 7), and generalization to unseen interactions (Fig. 8). All results demonstrate the superior performance of our method compared to prior work.

Different Viewpoints. Fig. 9 presents the visualization of the rendering results from different viewpoints, demon-

strating the robustness of our PhysTwin in handling various viewpoints.

Ablation Study on Hierarchical Optimization. To better understand the importance of our hierarchical sparse-to-dense optimization strategy, we conduct ablation studies with two variants: one using only zero-order optimization and the other using only first-order optimization. These experiments are performed on both the reconstruction & resimulation task and the future prediction task. Table 3 presents the results of different variants. Our complete pipeline achieves the best performance across both tasks. The variant with only zero-order optimization fails to capture fine-grained material properties, limiting its ability to represent different objects. On the other hand, the variant with only first-order dense optimization neglects the optimization of non-differentiable parameters, such as the spring connections. The default connections fail to accurately model the real object structure, and the connection distances between control points and object points cannot be effectively handled with a fixed initialization value.

Tracking Results. Fig. 6 shows the visualization of our tracking results and the pseudo-GT tracking results from CoTracker3 [23]. Even though our PhysTwin is optimized with noisy GT tracking, our model achieves much better and smoother tracking results during both the reconstruction & resimulation and future prediction tasks.

Data Efficiency Experiment. To further analyze the performance difference between our method and the GNN-based approach, we collected 29 additional data points on the same motion (double-hand stretching and folding rope), bringing the total to 30 data points for training the neural dynamics model. In contrast, our method is trained using only 1 data point. The results show that GS-Dynamics does not show a performance boost even with 30 times more data than our method. This indicates that their approach is data-hungry, whereas our method demonstrates significantly better data efficiency in learning a useful dynamics model. Even with 30 times more data, the learning-based method still struggles to capture precise dynamics as effectively as our approach.

D. Future Work

Our work takes an important step towards constructing an effective physical digital twin for deformable objects from sparse video observations. Unlike existing methods that primarily focus on geometric reconstruction, our approach integrates physical properties, enabling accurate resimulation, future prediction, and generalization to unseen interactions. Despite using three RGBD views in our current setup, our framework is inherently flexible and can extend to even sparser observations. With appropriate priors, a single RGB video could serve as a promising and scalable alternative, making our approach more applicable to in-the-wild scenarios. Furthermore, while our framework optimizes physical parameters based on a single type of interaction, expanding to multiple action modalities could further enhance the estimation of an object's intrinsic properties. Learning from a broader range of interactions may reveal richer physical characteristics and improve robustness. Beyond reconstruction and resimulation, our method opens up exciting possibilities for downstream applications, particularly in robotics. By providing a structured yet efficient digital twin, our approach significantly simplifies real-to-sim transfer, reducing the reliance on domain randomization for reinforcement learning. Additionally, the high-speed simulation and real-time rendering capabilities of our framework pave the way for more effective model-based robotic planning. By bridging the gap between perception and physics-based simulation, our method lays a solid foundation for future advancements in both computer vision and robotics.