2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

PLOP: Particle Filtering for Learning Object Physics from Robot Interaction Videos

Junyu Nan[1], Sergey Zakharov[2], Kris Kitani[1]

***Abstract*— Learning the dynamics of deformable objects, such as dough or a sponge, from RGB-D videos is challenging due to insufficient visual cues and complex deformations. We introduce PLOP (Particle Filtering for Learning Object Physics), a novel framework to learn the dynamics model of deformable objects using a particle filter over 3D Gaussians. Our method learns a dynamics function to predict the state of the object in the next time step, and a resampling function to split and merge Gaussians to handle complex object deformations such as cutting. We propose I2N (Implicit Particle Interaction Network) as the dynamics function within PLOP, a model leveraging a mixed particle-grid representation inspired by the Material Point Method (MPM). By transferring particle features to grid nodes, solving for grid dynamics, and then projecting solutions back to particles, our approach avoids the need for explicit pairwise interaction reasoning between particles, significantly reducing computational cost when there is a large number of particles. While PLOP is applicable to general robot-object interactions, we evaluate our approach on cutting sequences in both simulation and the real world, which induce challenging topological changes and expose previously occluded surfaces. On these benchmarks, PLOP achieves a 53.15% improvement in 3D reconstruction accuracy and a 6.84% improvement in 2D reconstruction accuracy on the simulation benchmark, as well as 28.41% and 24.45% improvements in 3D and 2D reconstruction metrics, respectively, on the real-world dataset.**

I. Introduction

Learning the dynamics model of deformable objects often relies on ground-truth particle trajectories as supervision [1], [2], [3]. However, particle trajectories can be challenging to obtain from real-world robot interaction videos, due to limited visual cues and complex deformations, especially for soft materials like dough or sponge. Previous work [3] models the deformable object by initializing a fixed set of 3D Gaussians [4] on the surface of the object, optimizing the trajectories of the surface particles, and then learning the dynamics model using the optimized particle trajectories as the ground truth. However, this approach cannot handle complex topological changes such as tearing or cutting, since surface particles after the deformation might be occluded at the beginning.

To address this limitation, we propose to represent the deformable object as an adaptive set of 3D Gaussians [4], and introduce PLOP (**P**article Filtering for **L**earning **O**bject **P**hysics) to learn a dynamics model over this adaptive representation through particle filtering. In our framework, the state estimate (a collection of particles) is propagated forward in time by a learned dynamics model and adaptively updated every time step through a learned resampling function. The resampling function predicts updated weights for each particle sample and then splits or merges Gaussians to adjust the set of particles. This resampling step allows our method to represent and predict complex topological changes, including cutting and tearing, which is difficult to handle using a fixed number of surface particles. The framework is illustrated in Figure 1.

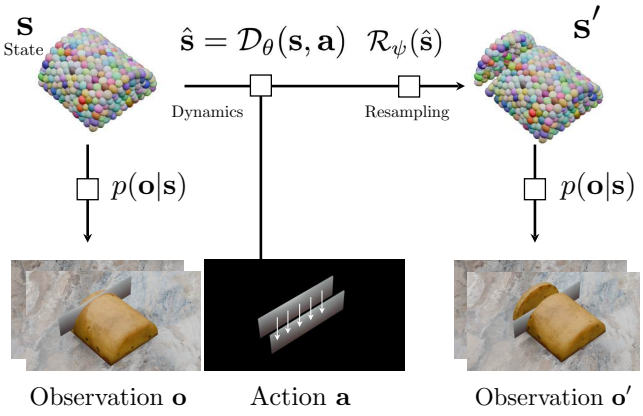


Fig. 1: PLOP models deformable objects as a mixture of Gaussians that transform over time according to a dynamics model and resampling process. States are estimated from multiview RGB-D observations of robot interaction with deformable objects.

As complex deformations may require many Gaussians to model, efficiency is crucial when designing the dynamics model. We propose I2N (**I**mplicit Particle **I**nteraction **N**etwork), a network using a mixed particle-grid representation inspired by the Material Point Method (MPM) [5] to improve computational efficiency. Unlike Graph Neural Networks (GNNs) [1], [6], which explicitly reason about pairwise interactions between particles, I2N propagates particle features to a grid, updates dynamics on grid nodes, and interpolates updates back to particles. As each particle only exchanges information with a constant number of grid nodes, I2N improves scalability for large particle sets compared to GNN-based methods [1], [6].

We focus on cutting because it is a canonical topology-changing interaction that directly stresses the need for adaptive resampling to model object splitting and newly exposed surfaces under occlusion. Our method is evaluated on both simulation (DiSECt [7]) and real-world (DROID-Deformable,

[1]Authors are with the Robotics Institute, Carnegie Mellon University. jnan1, kmkitani@andrew.cmu.edu
[2]Authors are with the Toyota Research Institute. sergey.zakharov@tri.global

collected with the DROID [8] setup) cutting benchmarks, outperforming the baseline [3] with 53.15% and 6.84% improvements in 3D and 2D reconstruction on simulation, and 28.41% and 24.45% on the real-world dataset.

Our main contributions are as follows: (1) We introduce a new representation for deformable objects using a dynamic set of 3D Gaussians to model topological changes. (2) We propose PLOP to learn the dynamics of this adaptive representation using a particle filter. (3) A new dynamics model, I2N, is proposed to predict particle dynamics efficiently with a mixed particle-grid representation. (4) A learnable resampling function is proposed to predict updated particle weights to control Gaussian splitting and merging and adjust the set of particles. (5) We validate our method by comparing it with previous work on simulation and real-world cutting benchmarks.

II. Related Works

Learning Dynamics from Robot Interaction Data is challenging due to the lack of particle correspondences over time. Early works optimize the dynamics model using indirect supervision, applying losses over point clouds [6], [9] or NeRF representations [10], [11]. However, as they rely purely on visual reconstruction without explicitly enforcing physically meaningful motion, these methods often struggle with textureless or occluded surfaces. [3] first tracks Gaussians under physical constraints to obtain physically plausible particle trajectories and then fits a dynamics model. Since the method assumes a fixed number of Gaussians, it cannot handle complex topological changes like cutting. Our method addresses this challenge through a particle filtering framework over 3D Gaussians, with a resampling step to adjust the weights of 3D Gaussians and dynamically merge or split Gaussians.

Tracking through Bayesian Filtering provides a probabilistic framework for recursive state estimation given noisy sensor observations [12], [13], [14]. Recent works adapt this idea for tracking 3D Gaussians by updating Gaussian parameters over time based on RGB-D observations [15], or estimating Gaussian deformations between frames for dynamic scene rendering and point tracking [16]. However, both methods assume a fixed set of Gaussians for tracking and do not integrate an explicit motion model. In contrast, our method learns a dynamics model to estimate state evolution using a particle filter, and introduces an adaptive resampling step that enables our approach to capture complex topological changes, such as material splitting.

Learning Particle Dynamics through Graph-based models [1], [17] and transformer-based approaches [2] often scale poorly with the number of particles. As a large number of Gaussians might be needed to represent deformable objects undergoing complex topological changes, we adopt a mixed particle-grid representation inspired by MPM [5] to achieve efficient particle dynamics learning. This representation eliminates explicit edge construction between particle pairs as in the GNN-based method [1], [17], improving scalability against a large number of particles.

III. Method

Our framework, PLOP, learns the dynamics of deformable objects using a particle filter over 3D Gaussians. At each time step, the particle filter updates the set of particles, namely the set of 3D Gaussians, by our dynamics model, I2N, which predicts the future state given the current state and the robot action. Then, a resampling function predicts the updated particle opacity and adjusts the set of particles by splitting and merging Gaussians, enabling a more flexible representation to handle large deformations. Repeating these steps over time leads to sequential state estimations that can be rendered as sensor observations. The entire framework is optimized so that the state estimations match the sensor observations and comply with physical constraints. An overview of the method is illustrated in Figure 2.

A. Problem Statement

Our goal is to learn a dynamics function $p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$ that maps the state of an object $\boldsymbol{s}$ to the next state $\boldsymbol{s}'$ given a robot action $\mathbf{a}$. To provide the technical basis for learning the dynamics function, we start by describing how we apply the Bayesian filtering framework to this problem.

We are given as input a sequence of observed interactions (*i.e.* , a robot interacting with a deformable object) $\mathcal{Q} = \{\mathbf{q}^i\}$, where each interaction consists of a tuple: a sensor observation $\mathbf{o}$ in the form of a set of multiview RGB-D images; robot action $\mathbf{a}$; and the resulting next observation $\mathbf{o}'$,

$$\mathbf{q}^i = \{(\mathbf{o}_t^i, \mathbf{a}_t^i, \mathbf{o}_{t+1}^i) \mid t = 0, \ldots, T^i\}. \tag{1}$$

Sensor observations $\mathbf{o}_t$ can be interpreted as a noisy measurement of the true state of the object $\boldsymbol{s}$, as they do not provide direct estimates of all the underlying physical properties of the object that determine its dynamics.

Within the framework of Bayesian filtering, the state estimation problem is solved by computing the posterior distribution over states given the history of observations and actions:

$$p(\mathbf{s}_t|\mathbf{o}_{0:t}, \mathbf{a}_{0:t-1}). \tag{2}$$

The computation of the posterior distribution can be decomposed into two recursive steps:

$$\textbf{Prediction:} \quad p(\mathbf{s}_t \mid \mathbf{o}_{t-1}, \mathbf{a}_{t-1}) = \int p(\mathbf{s}_t \mid \mathbf{s}_{t-1}, \mathbf{a}_{t-1})\, p(\mathbf{s}_{t-1} \mid \mathbf{o}_{t-1})\, d\mathbf{s}_{t-1}, \tag{3}$$

$$\textbf{Update:} \quad p(\mathbf{s}_t \mid \mathbf{o}_t) \propto p(\mathbf{o}_t \mid \mathbf{s}_t)\, p(\mathbf{s}_t \mid \mathbf{o}_{t-1}, \mathbf{a}_{t-1}). \tag{4}$$

where the *prediction* step is derived from marginalization, and the *update* step is obtained from Bayes' rule. Next, we will explain the components of this Bayesian filter.

B. State and Action Representation

We utilize a very high-dimensional state space and represent the state of a deformable object $\boldsymbol{s}$ by a set of 3D Gaussians

$$\mathbf{s}_t = \mathbf{G}_t = (\mathbf{X}_t, \mathbf{R}_t, \mathbf{S}_t, \mathbf{SH}_t, \boldsymbol{\sigma}_t), \tag{5}$$

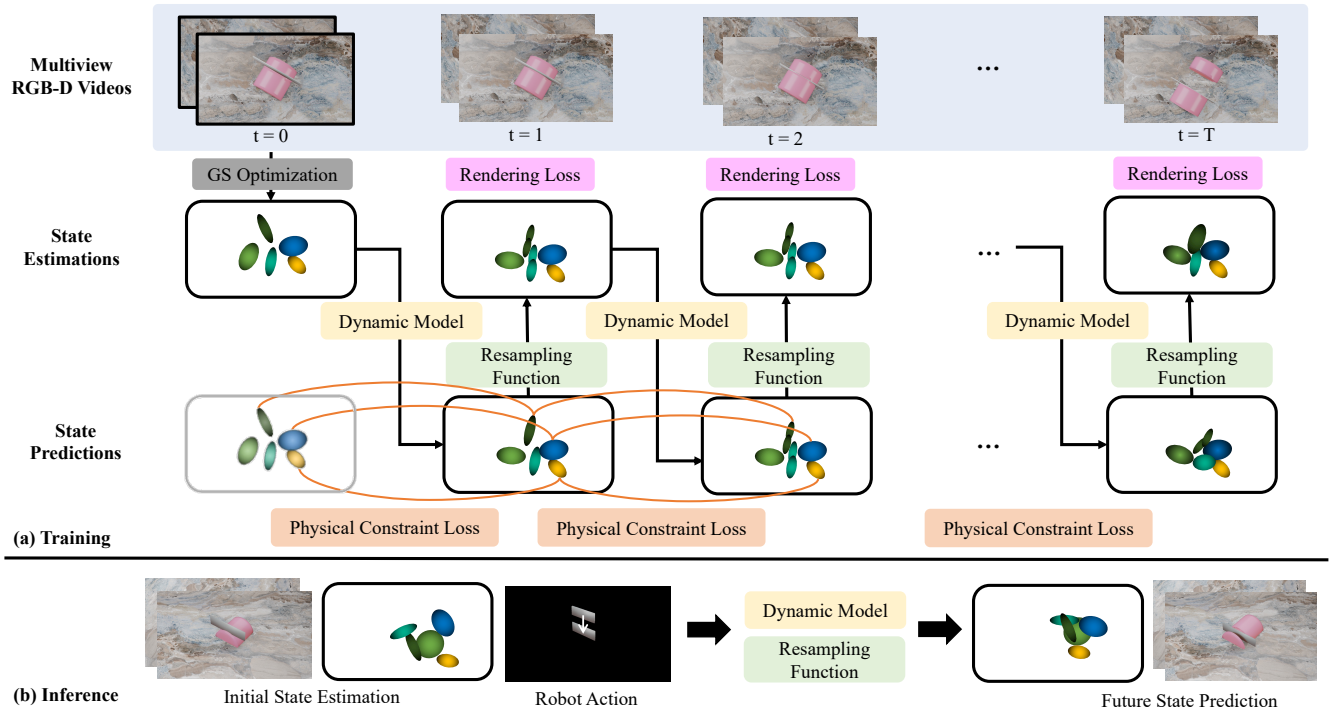


Fig. 2: Overview of our method. (a) At training time, our method takes in multi-view RGB-D videos, optimizes state representation at $t = 0$ against sensor observations through Gaussian Splatting, and then learns the dynamics model through particle filtering over 3D Gaussians. The optimization is supervised by rendering and physical constraints losses. (b) The learned dynamics model is evaluated at inference time by rolling out future states given an initial state and action sequence as input.

where $\mathbf{X}_t$ represents the mean position, $\mathbf{R}_t$ and $\mathbf{S}_t$ define the covariance matrices $\mathbf{\Sigma}_t = \mathbf{R}_t\mathbf{S}_t\mathbf{S}_t^T\mathbf{R}_t^T$, $\mathbf{SH}_t$ encodes the view-dependent appearance using spherical harmonics, and $\boldsymbol{\sigma}_t$ represents opacity. Note that the number of Gaussians in $\mathbf{G}_t$ may change over time steps through our proposed resampling step.

Similarly, the robot action $\boldsymbol{a}$ is represented by a set of Gaussians plus their motions:

$$\mathbf{a}_t = \mathbf{A}_t = (\mathbf{X}_t, \mathbf{R}_t, \mathbf{S}_t, \mathbf{SH}_t, \sigma_t, \Delta\mathbf{X}_t, \Delta\mathbf{R}_t, \Delta\mathbf{S}_t), \quad (6)$$

where $\Delta\mathbf{S}_t$ is enforced to be zero vectors since the actor is assumed to consist of rigid links. For example, in a cutting sequence, $\mathbf{A}_t$ will be a set of Gaussians reconstructing the blade that the robot holds, with shared $\Delta\mathbf{X}_t, \Delta\mathbf{R}_t$ over all Gaussians describing the cutting motion.

C. State Estimation using a Particle Filter

Given our state definition, the posterior distribution $p(\mathbf{s}_t|\mathbf{o}_{0:t}, \mathbf{a}_{0:t-1})$ is approximated by a mixture of Gaussians:

$$p(\mathbf{s}_t|\mathbf{o}_{0:t}, \mathbf{a}_{0:t-1}) \approx \sum_{i=1}^{N} \sigma_t^{(i)} \mathcal{N}(\mathbf{G}_t; \mathbf{X}_t^{(i)}, \mathbf{\Sigma}_t^{(i)}). \quad (7)$$

This approximation aligns with particle filtering in using point samples (particles) to approximate a probability density function. Hence, our method is a particle filter, where the 3D Gaussians represent the particles and their opacity $\boldsymbol{\sigma}_t$ corresponds to the importance weights that describe the contribution of each Gaussian to the state estimation.

At the initial time step $t = 0$, the particles $\mathbf{G}_0$ are initialized to match the sensor observations $\mathbf{o}_0$. Using the rendering process of Gaussian Splatting [4], denoted as a rendering function $\mathcal{H}$, the initial particles $\boldsymbol{G}_0$ are initialized by solving:

$$\mathbf{G}_0^\star = \arg\min_{\mathbf{G}} \mathcal{L}_{\text{render}}[\mathcal{H}(\mathbf{G}), \mathbf{o}], \quad (8)$$

where $\mathcal{L}_{\text{render}}$ is the rendering loss function in RGB-D image space:

$$\mathcal{L}_{\text{render}} = \lambda_{\text{SSIM}}\,\mathcal{L}_{\text{SSIM}} + \lambda_{\text{L1}}\,\mathcal{L}_{\text{L1}} + \lambda_{\text{depth}}\,\mathcal{L}_{\text{depth}}, \quad (9)$$

where λ_{SSIM}, λ_{L1} are weights for SSIM and L1 losses against ground-truth RGB images, and λ_{depth} is the weight of L1 loss against ground-truth depth images. In practice, we apply an additional anisotropic regularizer to prevent over-stretched Gaussians:

$$\mathcal{L}_{\text{aniso}} = \frac{1}{|\mathcal{G}|} \sum_{g\in\mathcal{G}} \max\left\{\max\left(\boldsymbol{S}_g\right) / \min\left(\boldsymbol{S}_g\right), r\right\} - r, \quad (10)$$

which penalizes a large ratio between the lengths of the major and minor axes to remain below a predefined threshold r.

We also perform an internal filling step at the 10k-th iteration following [18]: the bounding volume is discretized into a 16^3 opacity grid, and rays are cast along six axes; where a ray transitions from low-opacity ($\sigma < 0.1$) to high-opacity ($\sigma > 0.1$), a new Gaussian is placed with properties inherited from the nearest existing Gaussian.

Starting from $t = 1$, we apply the two-step state estimation process introduced in Eq. 3. Following the *prediction* step, the

particles are propagated to the next time step using a learnable dynamics function $\mathcal{D}_\theta(\mathbf{s}_{t-1}, \mathbf{a}_{t-1})$ designed to approximate the *one-step prediction* of particles under the true dynamics (motion) model.

$$\hat{\mathbf{G}}_t \sim p(\mathbf{s}_t|\mathbf{o}_{t-1}, \mathbf{a}_{t-1}) \tag{11}$$

$$= \int p(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_{t-1}) p(\mathbf{s}_{t-1}|\mathbf{o}_{t-1}) d\mathbf{s}_{t-1} \tag{12}$$

More details of the dynamics model are given in Section III-E.

The *update* step requires specification of a likelihood function (observation model) $p(\mathbf{o}_t|\mathbf{s}_t)$, which we approximate using the rendering loss from Eq. 9: $p(\mathbf{o}_t|\mathbf{s}_t) \propto -\mathcal{L}_{\text{render}}[\mathcal{H}(\mathbf{G_t}), \mathbf{o}]$.

In traditional particle filtering, a resampling step follows the update to prevent sample degeneracy. We learn a resampling model $\mathcal{R}_\psi$ to predict the Gaussian opacity update $\Delta\boldsymbol{\sigma}_{t-1}$, then adjust the Gaussian set based on the updated importance weights (details in Section III-F). Combining dynamics and resampling, the updated state is:

$$\mathbf{G}_t = \mathcal{R}_\psi(\mathcal{D}_\theta(\mathbf{G}_{t-1}, \mathbf{A}_{t-1})). \tag{13}$$

State estimation at any time step t can be obtained by recursively applying Eq. 13.

D. Optimization

Given Eq. 13, we optimize the dynamics parameters θ and resampling parameters ψ against the rendering loss (Eq. 9) and a physical constraint loss that ensures smooth, feasible motion:

$$\arg\min_{\theta,\psi} \sum_{t=1}^{T} \mathcal{L}_{\text{render}}[\mathcal{H}(\mathcal{R}_\psi(\mathcal{D}_\theta(\mathbf{G}_{t-1}, \mathbf{A}_{t-1})), \mathbf{o}_t] + \mathcal{L}^t_{\text{physical}}. \tag{14}$$

$\mathcal{L}_{\text{physical}}$ includes rigidity, rotational similarity, and long-term isometry terms [19]:

$$\mathcal{L}^t_{\text{physical}} = \lambda_r \mathcal{L}^t_{\text{rigid}} + \lambda_{\text{rot}} \mathcal{L}^t_{\text{rot}} + \lambda_i \mathcal{L}^t_{\text{iso}}, \tag{15}$$

where λ_r, λ_{rot}, λ_i are weights for rigidity loss $\mathcal{L}_{\text{rigid}}$, rotational similarity loss $\mathcal{L}_{\text{rot}}$, and isometry loss $\mathcal{L}_{\text{iso}}$ following the definitions in [19].

E. Dynamics Function: I2N

To learn the dynamics model $\mathcal{D}_\theta(\mathbf{G}, \mathbf{A})$ effectively over a large number of 3D Gaussians, we propose a new dynamics network using a mixed particle-grid representation inspired by MPM [5], a numerical technique used in fluid dynamics and soft-body physics simulation. We name our network **I**mplicit Particle **I**nteraction **N**etwork (I2N) as there is no explicit modeling of pairwise particle interaction. Figure 3 provides an overview of the dynamics model, with each component explained below.

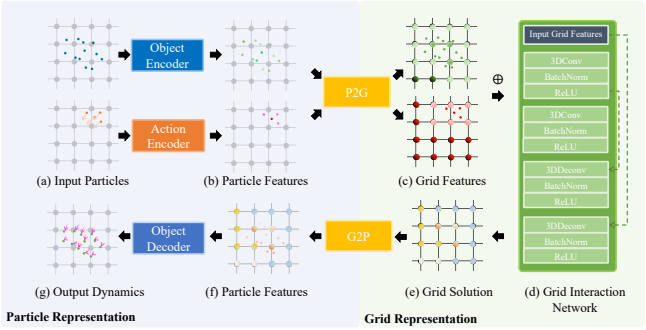


Fig. 3: Overview of I2N. **(a)** Input object and action particles. **(b)** Particle features encoded by object and action encoders. **(c)** Particle features propagated to grids using P2G operation. **(d)** Grid Interaction Network updates grid features to simulate grid interactions to obtain grid solutions in **(e)**. **(f)** Updated grid features are propagated back to particles using G2P operation. **(g)** Updated particle features are projected to particle dynamics space using an object decoder.

1) Particle Representation: Our dynamics model takes the Gaussians of the object $\mathbf{G}$ and the action $\mathbf{A}$ as input. Each input is regarded as a set of particles $\mathbf{P}$ with positions $\mathbf{X}$ and features $\mathbf{V}$. In this particle representation, each particle $\mathbf{p} = (\mathbf{x}, \mathbf{v})$, where $\mathbf{x}_p = (x_p, y_p, z_p)$ denotes the particle coordinates in Cartesian space, and $\mathbf{v}$ represents particle features obtained by encoding attributes of each object or action Gaussian. Object particle features $\mathbf{V}^g_t$ are obtained by encoding opacity σ, volume approximated by $\det(SS^T)$, as well as motion features from the last timestep $\Delta\mathbf{X}_{t-1}, \Delta\mathbf{R}_{t-1}$:

$$\mathbf{V}^g_t = f^g_{\text{enc}}(\sigma^g_t, \det(\mathbf{S}^g(\mathbf{S}^g)^T), \Delta\mathbf{X}^g_{t-1}, \Delta\mathbf{R}^g_{t-1}). \tag{16}$$

Action particle features $\mathbf{V}^a_t$ encode σ and $\det(SS^T)$ of the action Gaussians, plus $\Delta\mathbf{X}_t, \Delta\mathbf{R}_t$ to represent the action taken by the robot at the current time step:

$$\mathbf{V}^a_t = f^a_{\text{enc}}(\sigma^a_t, \det(\mathbf{S}^a(\mathbf{S}^a)^T), \Delta\mathbf{X}^a_t, \Delta\mathbf{R}^a_t). \tag{17}$$

2) Grid Representation: The grid is represented by a set of $M \times M \times M$ grid nodes, each with indices $\mathbf{i} = (i, j, k)$ where $i, j, k \in [1, M]$, Cartesian coordinates (ih, jh, kh) where h represents the grid spacing, and grid node features $\mathbf{N}^g_t$.

3) P2G and G2P Operations: Features are transferred back and forth between the grid and particle representation spaces through P2G and G2P operations. To compute the grid features $\mathbf{n_i}$ from particles $\mathbf{P}$, a projection weight is computed from each particle $\mathbf{p}$ to each grid node $\mathbf{i}$:

$$w_{\mathbf{i}}(\mathbf{x}_p) = \mathcal{K}(x_p - ih)\mathcal{K}(y_p - jh)\mathcal{K}(z_p - kh), \tag{18}$$

where $\mathcal{K}(x)$ is a cubic kernel [5] defined as

$$\mathcal{K}(x) = \begin{cases} \frac{1}{2}|\frac{x}{h}|^3 - (\frac{x}{h})^2 + \frac{2}{3}, & \text{for } 0 \le x < h \\ 0, & \text{otherwise} \end{cases}. \tag{19}$$

The P2G operation computes each grid node feature $\mathbf{n_i}$ as a weighted average of particle features $\mathbf{v}_p$:

$$\mathbf{n_i} = \sum_p \frac{w_\mathbf{i}(\mathbf{x}_p)}{\sum_p w_\mathbf{i}(\mathbf{x}_p)} \mathbf{v}_p, \tag{20}$$

while the G2P operation computes the particle features $\mathbf{v}_p$ as a weighted average of the grid features $\mathbf{n_i}$:

$$\mathbf{v}_p = \sum_\mathbf{i} \frac{w_\mathbf{i}(\mathbf{x}_p)}{\sum_\mathbf{i} w_\mathbf{i}(\mathbf{x}_p)} \mathbf{n_i}. \tag{21}$$

4) Particle Dynamics Representation: The updated particle features are then projected by f_{dec} into the particle update space $\Delta\mathbf{G}_t$ parameterized by $(\Delta\mathbf{X}_t, \Delta\mathbf{R}_t, \Delta\mathbf{S}_t)$:

$$\hat{\mathbf{G}}_t = \mathbf{G}_t \oplus \Delta\mathbf{G}_t = (\mathbf{X}_t + \Delta\mathbf{X}_t, \Delta\mathbf{R}_t \cdot \mathbf{R}_t, \mathbf{S}_t + \Delta\mathbf{S}_t, \mathbf{SH}, \boldsymbol{\sigma}_t)$$

where spherical harmonics $\mathbf{SH}$ is consistent over time, and $\Delta\mathbf{X}_t, \Delta\mathbf{R}_t, \Delta\mathbf{S}_t$ correspond to the motion of particles as a result of robot action in the *prediction* step of particle filtering.

5) Network Architecture: The network architecture is illustrated in Figure 3. Given input object and action Gaussians in particle representations, *i.e.* , $(\mathbf{X}_t^g, \mathbf{V}_t^g)$ and $(\mathbf{X}_t^a, \mathbf{V}_t^a)$, where the particle features are extracted by encoders through Eq. 16 and Eq. 17, $\Delta\mathbf{G}_t$ is predicted by the following steps:

$$\mathbf{N}_t^g = \text{P2G}(\mathbf{V}_t^g, \mathbf{X}_t^g), \quad \mathbf{N}_t^a = \text{P2G}(\mathbf{V}_t^a, \mathbf{X}_t^a) \tag{22}$$

$$\mathbf{N}_{t+1}^g = f_{\text{grid}}(\mathbf{N}_t^g, \mathbf{X}_t^a), \quad \mathbf{V}_{t+1}^g = \text{G2P}(\mathbf{N}_{t+1}^g, \mathbf{X}_t^g) \tag{23}$$

$$\Delta\mathbf{G}_t = f_{\text{dec}}(\mathbf{V}_{t+1}^g, \mathbf{V}_t^g). \tag{24}$$

Particle encoders f_{enc}^g, f_{enc}^a and decoder f_{dec} are MLPs, and f_{grid} is implemented by 3D convolution and deconvolution layers as shown in Figure 3(d).

6) Complexity Analysis: Given Eq. 19, the weights $w_\mathbf{i}(\mathbf{x}_p)$ only need to be computed between the 8 closest grid nodes for each particle $\mathbf{p}$. Therefore, the computational complexity of the P2G and G2P operations is $O(M^3 + N)$, where M is the grid dimension and N is the number of particles. In comparison, previous dynamics models [1], [2] need to model direct interaction between particle pairs, which is of complexity $O(N^2)$. As a result, our method is more efficient when the number of particles is large while the grid dimension is reasonably small.

F. Resampling Function

Resampling prevents degeneracy in particle filtering by discarding low-weight and duplicating high-weight particles. Our resampling function $\mathcal{R}_\psi$ consists of three components: a model $\mathcal{R}_\psi^\sigma$ to predict updated particle weights, a splitting operator $\mathcal{R}^s$, and a merging operator $\mathcal{R}^m$:

$$\mathbf{G}_t = \mathcal{R}_\psi(\mathcal{D}_\theta(\mathbf{G}_{t-1}, \mathbf{A}_{t-1})) \tag{25}$$

$$= \mathcal{R}^s \circ \mathcal{R}^m \circ \mathcal{R}_\psi^\theta(\mathcal{D}_\sigma(\mathbf{G}_{t-1}, \mathbf{A}_{t-1})) \tag{26}$$

$\mathcal{R}_\psi^\sigma$ shares the same architecture and weights with the dynamics model $\mathcal{D}_\theta$, except a different output head $f_{dec}^\sigma(\mathbf{V}_{t+1}^g, \mathbf{V}_t^g)$ to predict $\Delta\boldsymbol{\sigma}_{t-1}$ at each time step t. $\mathcal{R}^m$ and $\mathcal{R}^s$ are nonparametric functions for merging and splitting Gaussians based on their opacity and covariance.

1) Merging Operation ($\mathcal{R}^m$): To prevent excessive Gaussians with low weights, we merge Gaussians with low opacity into their closest surviving neighbor. A Gaussian $\mathbf{G}_i$ is considered for merging if $\sigma_t^i < \tau_m$, where τ_m is an opacity threshold. The merging process identifies the closest surviving Gaussian $\mathbf{G}_j$ measured in Euclidean distance and updates its parameters as follows:

$$\mathbf{X}_j = \frac{\sigma_i \mathbf{X}_i + \sigma_j \mathbf{X}_j}{\sigma_i + \sigma_j}, \quad \Sigma_j = \frac{\sigma_i \Sigma_i + \sigma_j \Sigma_j}{\sigma_i + \sigma_j}, \quad \sigma_j = \sigma_i + \sigma_j. \tag{27}$$

The Gaussian $\mathbf{G}_i$ is then removed from the state representation.

2) Splitting operation ($\mathcal{R}^s$): We split Gaussians with high opacity $\hat{\sigma}^{(i)} >= \tau_m$ and highly elongated covariance matrix. For each Gaussian $\mathbf{G}_i$, we compute the ratio of its maximum to minimum eigenvalue by $r_i = \frac{\mathbf{S}_{max}^{(i)}}{\mathbf{S}_{min}^{(i)}}$, and select it for splitting if $r_i > \tau_s$, where τ_s is a threshold controlling the sensitivity of the splitting process.

We apply the method proposed by [20] to split a selected Gaussian along its principal axes. Given the eigenvalue decomposition of the covariance matrix $\boldsymbol{\Sigma}_i$, the splitting direction is chosen along the eigenvector $\mathbf{e}_{\max}$ corresponding to the largest scaling value $\mathbf{S}_{max}^{(i)}$, and the new Gaussians are placed symmetrically at a displacement $v\mathbf{e}_{\max}$ from the original mean $\mathbf{X}_i$:

$$\mathbf{X}_1 = \mathbf{X}_i + v\mathbf{e}_{\max}, \quad \mathbf{X}_2 = \mathbf{X}_i - v\mathbf{e}_{\max}. \tag{28}$$

Both new components share the same variance adjusted based on the displacement $\Sigma_1 = \Sigma_2 = (1 - v^2)\Sigma_i$. The mixture weights are set to 0.5 each, ensuring the total probability mass remains unchanged. The displacement parameter v is randomly sampled within the range $[-1, 1]$.

IV. Experiments

To validate our framework, we evaluated our method against the baseline [3] on both simulation and real-world cutting benchmarks. We use [3] as the primary baseline because most prior dynamics-learning approaches are not directly comparable in our end-to-end setting: they either require ground-truth particle trajectories or non-visual state supervision, or they do not produce rollout-and-render predictions from RGB-D videos that match our evaluation protocol.

A. Datasets

DiSECt [7] Dataset contains cutting sequences from 9 environments simulated using DiSECt [7]. The simulator is run for each environment with 30 different initial cutting positions and speeds, generating USD files, which are then rendered using Blender. Figure 4 contains a sample frame from each of the 9 environments, corresponding to different combinations of geometries (sphere, prism, cylinder) and materials (polymer, foam, jelly).

DROID-Deformable is a real-world dataset we collected using the DROID setup [8]. The setup contains 3 RGB-D cameras, where two of them are static, and the third is attached to the gripper. Before each interaction sequence, the

Environment	Methods	CD ↓	EMD ↓	SSIM ↑	PSNR ↑	LPIPS ↓	$\mathcal{J}$ score ↑
Prism-Jello	gs-dynamics [3]	0.3015	7.0847	0.8706	23.7118	0.1977	0.9170
	ours	**0.2455**	**5.3936**	**0.8753**	**24.1135**	**0.1853**	**0.9461**
Prism-Foam	gs-dynamics [3]	1.5294	3.0361	0.8983	27.9460	0.2359	0.9504
	ours	**0.1249**	**2.2621**	**0.9030**	**28.1894**	**0.2179**	**0.9669**
Prism-Polymer	gs-dynamics [3]	0.3341	**2.7810**	0.8157	24.4632	0.2346	0.8858
	ours	**0.2738**	3.3405	**0.8466**	**25.6403**	**0.2189**	**0.9171**
Sphere-Jello	gs-dynamics [3]	0.1276	3.0097	0.8427	28.8201	0.3087	**0.9630**
	ours	**0.1061**	**2.6418**	**0.8792**	**30.9270**	**0.2984**	0.9628
Sphere-Foam	gs-dynamics [3]	0.1097	**1.5364**	0.8503	28.8595	0.3409	0.9710
	ours	**0.0914**	1.5532	**0.8566**	**29.7277**	**0.3390**	**0.9716**
Sphere-Polymer	gs-dynamics [3]	**0.0720**	1.1390	0.8568	30.0475	0.3662	0.9588
	ours	0.0738	**1.0462**	**0.8644**	**31.6364**	**0.3586**	**0.9613**
Cylinder-Jello	gs-dynamics [3]	0.6824	18.6796	0.7373	17.0356	0.3783	0.7319
	ours	**0.1941**	**5.008**	**0.8102**	**19.6128**	**0.2823**	**0.8411**
Cylinder-Foam	gs-dynamics [3]	0.4307	10.1680	0.8048	19.4233	0.3475	0.8765
	ours	**0.1647**	**5.3919**	**0.8597**	**22.2658**	**0.2511**	**0.9007**
Cylinder-Polymer	gs-dynamics [3]	0.4307	9.9456	0.7641	17.1613	0.3467	0.8236
	ours	**0.1888**	**6.2248**	**0.8502**	**21.3708**	**0.2739**	**0.9262**

TABLE I: Quantitative results on DiSECt [7] environments. We compare against gs-dynamics [3], the closest prior method that supports end-to-end evaluation via rollout-and-render from RGB-D observations. CD and EMD losses are in millimeters.

Fig. 4: Visualization of sample frames from training sequences from each of 9 environments generated using DiSECt [7].

robot is controlled to perform a scanning operation to capture multiview images of the object using the gripper camera. We use images from this scanning operation to reconstruct a complete object cloud, which is used to initialize the state estimation at $t = 0$. Since the gripper camera is often heavily occluded by the tool the robot is holding during the interaction, we only use images from the other two static cameras as supervision at the other time steps.

B. Implementation Details and Hyperparameters

1) Initial State Estimation: We estimate the initial state $\mathbf{s}_0$ by optimizing a standard 3D Gaussian Splatting model [4] for 20k steps with the anisotropic regularizer described in Sec. 3.3 and an internal filling step following [18]. The hyperparameters are: $\lambda_{\text{aniso}} = 0.1$, $r = 2.5$, $\lambda_{\text{SSIM}} = 0.1$, $\lambda_{\text{L1}} = 0.9$, $\lambda_{\text{depth}} = 0.2$ on DiSECt [7], and $\lambda_{\text{aniso}} = 0.1$, $r = 3$, $\lambda_{\text{SSIM}} = 0.1$, $\lambda_{\text{L1}} = 0.9$, $\lambda_{\text{depth}} = 0.05$ on DROID-Deformable.

2) Training Details: On both DiSECt [7] dataset and DROID-Deformable, we optimize Eq. 14 over all interaction sequences using shared settings: 2D reconstruction weights ($\lambda_{\text{SSIM}} = 0.1$, $\lambda_{\text{L1}} = 0.9$), physical constraint weights ($\lambda_r = 1$, $\lambda_{\text{rot}} = 0.05$), and learning rate 10^{-4}. For dataset-specific hyperparameters, DiSECt dataset uses $\lambda_{\text{depth}} = 0.2$, $\lambda_i = 1$, grid spacing $h = 0.05$, and 50k epochs, while DROID-Deformable uses $\lambda_{\text{depth}} = 0.05$, $\lambda_i = 2$, grid spacing $h = 0.02$, and 10k epochs. We reduce λ_{depth} on real data to be less sensitive to RGB-D sensor noise, and use a finer grid (smaller h) to match the smaller physical scale of real-world objects; we also increase λ_i to more strongly regularize long-term isometry under noisier observations.

C. Metrics

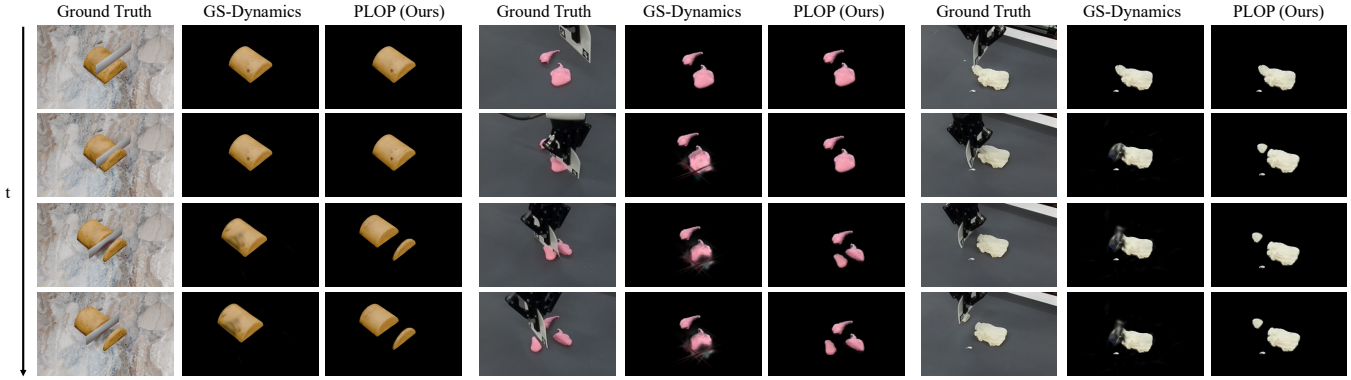
All experiments are evaluated by rolling out state predictions of our method and the baseline from an initial state and a sequence of actions unseen during training time, as illustrated in Figure 2(b). The state predictions are then rendered into RGB-D images and compared against the ground-truth observations using 3D reconstruction metrics (Chamfer Distance [21], Earth Mover's Distance [22]) and 2D reconstruction metrics (SSIM, PSNR, LPIPS [23], $\mathcal{F}$-score computed over masks). To speed up the computation, the point clouds are downsampled to 1024 points before computing the EMD loss [22]. To exclude the impact of the background, prediction and ground-truth images and masks are cropped to the bounding box of the unioned mask before computing the 2D reconstruction metrics.

D. Results and Analysis

We provide quantitative results on 9 cutting environments simulated using DiSECt [7] in Table I. The results indicate that our method significantly outperforms the baseline [3] in

Environment	Methods	CD ↓	EMD ↓	SSIM ↑	PSNR ↑	LPIPS ↓	$\mathcal{J}$ score ↑
Pink Slime	gs-dynamics [3]	0.5751	15.0119	0.5202	11.1192	0.4287	0.3916
	ours	**0.4443**	**9.439**	**0.5421**	**13.262**	**0.3944**	**0.6635**
White Kinetic Sand	gs-dynamics [3]	0.2344	9.5831	0.3683	9.3003	0.4695	0.6112
	ours	**0.1739**	**6.9938**	**0.5190**	**10.6089**	**0.3059**	**0.7348**

TABLE II: Quantitative results on DROID-Deformable environments. We compare against gs-dynamics [3], the closest prior method that supports end-to-end evaluation via rollout-and-render from RGB-D observations. CD and EMD losses are in millimeters.



(a) DiSECt (Cylinder-Polymer) (b) Pink Slime (c) White Clay

Fig. 5: Visualization of our method and [3] on **(a)** Cylinder-Polymer environment simulated by DiSECt [7], **(b)** pink slime and **(c)** white kinetic sand cutting interactions from DROID-Deformable.

handling topological changes induced by cutting. Figure 5(a) presents qualitative visualizations of model rollouts from both methods, demonstrating that our method learns the dynamics of cutting interactions much more accurately than the baseline, with a 53.15% improvement in 3D reconstruction losses and a 6.84% improvement in 2D reconstruction losses. We believe that the main limitation of the baseline [3] is representing the object using a fixed number of Gaussians with persistent shape and opacity. In contrast, our model dynamically adapts the Gaussian set by adjusting the weights and introducing new Gaussians through the resampling step, allowing it to predict previously occluded surfaces after cutting objects into multiple pieces.

Quantitative results on DROID-Deformable are presented in Table II, where our method consistently outperforms the baseline [3] on both objects across all metrics, with a 28.41% improvement in 3D reconstruction losses and a 24.45% improvement in 2D reconstruction losses. The losses are generally larger on DROID-Deformable compared to DiSECt [7], mainly due to sparser viewpoints, noisier sensor observations, and heavier occlusions. Figure 5 presents qualitative results confirming the same trend.

E. Ablations

1) Particle Dynamics Learning: To further analyze the effectiveness of our dynamics model I2N, we evaluate it against GNN [1] on the real-world dataset RoboCook [6] and the simulation dataset DPI-Net [1]. Note that in this section, the input and output are direct state estimations instead of sensor observations; as a result, only the dynamics model in PLOP is evaluated.

Results in Tables III and IV show our method outperforms GNN [1] in most environments while being faster than all of them.

Environment	Methods	CD ↓		EMD ↓		HD ↓		Time(ms) ↓
		MSE	FDE	MSE	FDE	MSE	FDE	
sym-rod	DPI-Net [1], [6]	3.680	4.779	2.394	3.171	9.662	11.438	11.008
	I2N (ours)	3.708	4.961	2.417	3.340	**9.487**	12.441	**8.956**
sym-plane	DPI-Net [1], [6]	3.970	6.206	2.698	4.409	11.004	17.381	6.672
	I2N (ours)	**3.866**	**6.192**	**2.679**	**4.663**	**10.521**	**15.573**	**4.944**
press-square	DPI-Net [1], [6]	3.709	4.924	2.446	3.457	9.628	12.906	7.372
	I2N (ours)	**0.801**	**4.800**	**0.520**	**3.119**	**2.005**	**11.774**	**5.236**
large-roller	DPI-Net [1], [6]	3.743	7.180	2.689	5.676	9.155	19.605	8.264
	I2N (ours)	**3.270**	**6.181**	**2.315**	**5.028**	**7.757**	**16.056**	**7.381**
small-roller	DPI-Net [1], [6]	3.740	7.420	3.211	7.504	12.990	26.663	7.736
	I2N (ours)	**2.976**	**5.819**	**2.028**	**4.539**	**7.221**	**15.111**	**4.576**
punch-square	DPI-Net [1], [6]	**0.536**	4.879	**0.338**	3.139	**1.223**	11.120	7.218
	I2N (ours)	0.785	**4.646**	0.497	**2.941**	1.743	**10.273**	**4.478**
asym-gripper	DPI-Net [1], [6]	**3.238**	6.893	2.164	4.709	**8.326**	19.391	8.467
	I2N (ours)	3.242	**6.505**	2.149	**4.419**	8.501	**16.846**	**4.982**

TABLE III: Quantitative results on RoboCook [6] environments. Numbers in magnitude of 10^{-3}.

	FluidFall		FluidShake	
	L2 ↓	Time (ms) ↓	L2 ↓	Time (ms) ↓
DPI-Net [1]	1.156 ± 0.635	21.6	$\mathbf{22.912 \pm 23.86}$	13.057
I2N (ours)	$\mathbf{0.707 \pm 0.110}$	**2.441**	23.432 ± 5.186	**5.996**

	RiceGrip		BoxBath	
	L2 ↓	Time (ms) ↓	L2 ↓	Time (ms) ↓
DPI-Net [1]	5.795 ± 5.136	26.5	18.995 ± 0.997	19.67
I2N (ours)	$\mathbf{3.338 \pm 1.731}$	**8.133**	$\mathbf{14.090 \pm 0.922}$	**4.390**

TABLE IV: Evaluation on DPI-Net environments [1] for the particle dynamics prediction task. Both I2N and DPI-Net are trained and evaluated over particle trajectories instead of RGB-D images.

2) Computational Efficiency: Table V reports how forward time scales with particle count on the large roller environment from RoboCook [6]. DPI-Net [1] runs out of memory beyond 5k particles due to explicit edge modeling, while I2N remains near-constant (7.4–11.3ms up to 10k particles).

Number of Particles	Time (ms) ↓	
	DPI-Net [1]	I2N (ours)
300	8.231	**7.381**
1k	25.564	**7.790**
2k	42.822	**8.680**
5k	Out of memory	**9.220**
10k	Out of memory	**11.332**

TABLE V: Scaling of I2N and DPI-Net [1] with respect to number of particles.

F. Limitations and Future Work

While PLOP improves performance across most environments and metrics, Table I shows a few cases with limited gains or small regressions, which can arise when cutting produces thin fragments or subtle high-frequency deformations; small geometric misalignments then make split/merge decisions more ambiguous. More broadly, our optimization depends on sufficient visual evidence and can degrade under severe occlusion or adverse sensing conditions (e.g., during robot pinching), where RGB-D signals become sparse, noisy, or unreliable. A promising future direction is to incorporate additional sensing modalities, especially tactile feedback, to provide complementary contact and deformation cues when vision fails, enabling more robust resampling and state estimation in challenging real-world interactions.

V. Conclusions

We introduced PLOP, a novel framework for learning deformable object dynamics from multiview RGB-D interaction videos using a particle filter over 3D Gaussians. By incorporating adaptive resampling, our method dynamically adjusts the number of Gaussians to track complex deformations and topological changes, ensuring a flexible and expressive representation. To efficiently model the evolving set of Gaussian particles, our proposed I2N adopts a mixed particle-grid representation inspired by the Material Point Method (MPM) to reduce computational cost over a large number of particles. Evaluations on simulation and real-world datasets demonstrate our method's superior performance over the baseline on challenging cutting scenes.

References

[1] Y. Li, J. Wu, R. Tedrake, J. B. Tenenbaum, and A. Torralba, "Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids," *ArXiv*, vol. abs/1810.01566, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:52917627

[2] Y. Shao, C. C. Loy, and B. Dai, "Transformer with implicit edges for particle-based physics simulation," in *European Conference on Computer Vision*, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:251018457

[3] M. Zhang, K. Zhang, and Y. Li, "Dynamic 3d gaussian tracking for graph-based neural dynamics modeling," in *8th Annual Conference on Robot Learning*, 2024. [Online]. Available: https://openreview.net/forum?id=itKJ5uu1gW

[4] B. Kerbl, G. Kopanas, T. Leimkuehler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics (TOG)*, vol. 42, pp. 1 – 14, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:259267917

[5] D. Sulsky, S.-J. Zhou, and H. L. Schreyer, "Application of a particle-in-cell method to solid mechanics," *Computer Physics Communications*, vol. 87, no. 1, pp. 236–252, 1995, particle Simulation Methods. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0010465594001707

[6] H. Shi, H. Xu, S. Clarke, Y. Li, and J. Wu, "Robocook: Long-horizon elasto-plastic object manipulation with diverse tools," *ArXiv*, vol. abs/2306.14447, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:259251806

[7] E. Heiden, M. Macklin, Y. S. Narang, D. Fox, A. Garg, and F. Ramos, "DiSECt: A Differentiable Simulation Engine for Autonomous Robotic Cutting," in *Proceedings of Robotics: Science and Systems*, Virtual, July 2021.

[8] A. K. et al., "Droid: A large-scale in-the-wild robot manipulation dataset," *ArXiv*, vol. abs/2403.12945, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:268531351

[9] H. Shi, H. Xu, Z. Huang, Y. Li, and J. Wu, "Robocraft: Learning to see, simulate, and shape elasto-plastic objects with graph networks," *ArXiv*, vol. abs/2205.02909, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:248562698

[10] Y. Li, S. Li, V. Sitzmann, P. Agrawal, and A. Torralba, "3d neural scene representations for visuomotor control," in *Conference on Robot Learning*, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:235765661

[11] D. Driess, Z. Huang, Y. Li, R. Tedrake, and M. Toussaint, "Learning multi-object dynamics with compositional neural radiance fields," *ArXiv*, vol. abs/2202.11855, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:247084284

[12] R. E. Kálmán, "A new approach to linear filtering and prediction problems" transaction of the asme journal of basic," 1960. [Online]. Available: https://api.semanticscholar.org/CorpusID:259115248

[13] P. Sarkar, "Sequential monte carlo methods in practice," *Technometrics*, vol. 45, pp. 106 – 106, 2003. [Online]. Available: https://api.semanticscholar.org/CorpusID:7923659

[14] M. S. Arulampalam, S. Maskell, N. J. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Trans. Signal Process.*, vol. 50, pp. 174–188, 2002. [Online]. Available: https://api.semanticscholar.org/CorpusID:55577025

[15] N. V. Keetha, J. Karhade, K. M. Jatavallabhula, G. Yang, S. Scherer, D. Ramanan, and J. Luiten, "Splatam: Splat, track & map 3d gaussians for dense rgb-d slam," *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21 357–21 366, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:265609060

[16] J. Lyu, J. Dong, and Y.-X. Wang, "LayeredGS: Efficient dynamic scene rendering and point tracking with multi-layer deformable gaussian splatting," 2024. [Online]. Available: https://openreview.net/forum?id=AkufxLzcV5

[17] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. W. Battaglia, "Learning to simulate complex physics with graph networks," *ArXiv*, vol. abs/2002.09405, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:211252550

[18] T. Xie, Z. Zong, Y. Qiu, X. Li, Y. Feng, Y. Yang, and C. Jiang, "Physgaussian: Physics-integrated 3d gaussians for generative dynamics," *ArXiv*, vol. abs/2311.12198, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:265309217

[19] J. Luiten, G. Kopanas, B. Leibe, and D. Ramanan, "Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis," *2024 International Conference on 3D Vision (3DV)*, pp. 800–809, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:261030923

[20] F. Faubel and D. Klakow, "Further improvement of the adaptive level of detail transform: Splitting in direction of the nonlinearity," *2010 18th European Signal Processing Conference*, pp. 850–854, 2010. [Online]. Available: https://api.semanticscholar.org/CorpusID:14094406

[21] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3d object reconstruction from a single image," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2463–2471, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:6746759

[22] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International Journal of Computer Vision*, vol. 40, pp. 99–121, 2000. [Online]. Available: https://api.semanticscholar.org/CorpusID:14106275

[23] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.