Supplementary Materials

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

Contents

1 Implementation Details **1**
1.1 Dynamic 3D Gaussian Tracking 1
1.2 Action-Conditioned Video Prediction 3
1.3 Model-Based Planning with MPPI 3
1.4 Real-World Experiments 4

2 Additional Results **4**
2.1 Tracking with Dynamic 3D Gaussian 4
2.2 Action-Conditioned Video Prediction 5
2.3 Model-Based Planning 6

3 Computational Complexity of Planning **7**

4 Ablation Study **7**
4.1 Objectives for Dynamic 3D Gaussian Tracking 7
4.2 Deploying Graph-Based Dynamics Model on Gaussians 8

5 Additional Limitation Discussions **8**
5.1 Limitation of Tracking Performance 8
5.2 Training Time of Dyn3DGS Tracking. 9
5.3 Generalization Capability of Dynamics Model 9

1 Implementation Details

1.1 Dynamic 3D Gaussian Tracking

Physics-inspired Optimization Objectives. We follow these physics-inspired optimization objectives of Dyn3DGS [1]:

The local rigidity objective $\mathcal{L}_{\text{rigid}}$ ensures that each Gaussian i and its neighbors j move congruently under rigid-body transformation across timesteps:

$$\mathcal{L}_{\text{rigid}}(i,j) = w_{i,j} \cdot \left\| \left(\mu_j^{t-1} - \mu_i^{t-1}\right) - R_i^{t-1} \left(R_i^t\right)^{-1} \left(\mu_j^t - \mu_i^t\right) \right\|_2, \tag{1}$$

$$\mathcal{L}_{\text{rigid}} = \frac{1}{k|S|} \sum_{i \in S} \sum_{j \in \text{knn}_{i;k}} \mathcal{L}_{\text{rigid}}(i,j). \tag{2}$$

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

The isometry objective maintains consistent distances between Gaussians across timesteps:

$$\mathcal{L}_{\text{isometry}}^{t} = \frac{1}{k|S|} \sum_{i \in S} \sum_{j \in \text{knn}_{i;k}} w_{i,j} \left| \left\| \mu_j^0 - \mu_i^0 \right\|_2 - \left\| \mu_j^t - \mu_i^t \right\|_2 \right|. \tag{3}$$

The rotation similarity objective ensures smooth rotational behavior among neighboring Gaussians:

$$\mathcal{L}_{\text{rotation}}^{t} = \frac{1}{k|S|} \sum_{i \in S} \sum_{j \in \text{knn}_{i;k}} w_{i,j} \left\| \hat{q}_j^{t-1} \odot (\hat{q}_i^{t-1})^{-1} - \hat{q}_i^t \odot (\hat{q}_j^t)^{-1} \right\|_2, \tag{4}$$

where $\odot$ denotes the quaternion product, and $(\hat{q}_i^t)^{-1}$ is the inverse quaternion for Gaussian i at timestep t. For all objectives, μ_i^t represents the Gaussian centers in 3D space, R_i^t denotes the rotation, and S is the set of all Gaussians. The weights $w_{i,j}$ are given by $w_{i,j} = \exp(-\lambda_w \|\mu_j^0 - \mu_i^0\|^2)$, where λ_w is a decay parameter, and $\mu_{i,0}$ and $\mu_{j,0}$ are the initial Gaussian positions.

Multi-Objective Balance. To account for the distinct physical properties of different objects, which are crucial for tracking optimization, we assign specific weights to each object category. We set $\lambda_{\text{rigid}} = 200$ for rope and toy animals, and $\lambda_{\text{rigid}} = 400$ for cloth. Additionally, we consider 20 Gaussian neighbors for the KNN of rope and toy animals, and 10 for the KNN of cloth to accurately reflect their unique characteristics. For the isometry objective, we set $\lambda_{\text{iso}} = 1000$ for rope and toy animals, and $\lambda_{\text{iso}} = 2000$ for cloth. The weights for other objectives remain consistent across different objects and instances.

Background Objective. To efficiently isolate and optimize dynamic components, we employ GroundingDINO [2] and Segment Anything [3] models to obtain masks for the objects the robot interacts with. This enhances optimization precision and efficiency. For the original Dyn3DGS, a foreground/background mask is rendered to increase scene contrast. They apply a background segmentation loss against a pseudo-ground-truth background mask, obtained by differencing an image without foreground objects. Additionally, a loss is applied to keep background points static, while rigidity, rotation, and isometry losses are restricted to foreground points. This improves efficiency and prevents enforcement between foreground objects and the static floor.

In their implementation, a "floor loss" prevents Gaussians from going below the floor, given the known ground plane of the scenes. In our configuration, we focus on optimizing the Gaussians of the objects and ignore the floor loss, as it does not contribute to tracking performance. However, we retain the background loss to ensure Gaussian points on the objects do not float into the background.

The background loss consists of two main components. The first measures the L1 loss between the current positions of the background points and their initial positions, penalizing significant deviations and ensuring they remain relatively stationary. The second measures the L1 loss between the current rotations of the background points and their initial rotations, ensuring their orientations do not change drastically and maintaining spatial consistency. Together, these components ensure that background points and their rotations remain close to their initial states, preserving the integrity of the background and preventing unnecessary adjustments that could interfere with the accurate modeling of dynamic foreground objects.

Opacity Filtering in Adaptive Densification. 3D Gaussian Splatting employs an adaptive densification scheme, involving the cloning and splitting of Gaussians, to regulate their quantity and density per unit volume. This process transitions from a sparse to a densely populated Gaussian set, enhancing scene representation accuracy. Low-opacity Gaussians contribute minimally to object appearance modeling, making it inefficient to optimize their transformations for high-quality tracking. Therefore, we retain only high-opacity Gaussians for dense correspondence.

Specifically, during the densification scheme, the density of points in a scene is adjusted based on various conditions and thresholds to optimize scene representation over time for the initialization of Gaussians. For up to 5000 iterations, gradients are accumulated to identify points needing adjustment. Every 100 iterations starting from the 500th, points are cloned based on their gradient values

and sizes. Some points are split into two, adjusted using normally distributed samples, and relevant variables are reset. Points to be removed are identified based on their opacity values and sizes, with additional removals after iteration 3000 to maintain balance.

Initially, the opacity of each point is evaluated using a sigmoid function. Points with opacity below a specified threshold are marked for removal. If the iteration count is 3000 or more, "big points" are identified based on their scale values, specifically comparing the maximum scale value of each point to 10% of the scene's radius. Points exceeding this size threshold are marked for removal. Finally, the list of points to be removed is updated, combining points identified by opacity and size criteria, ensuring an optimized and balanced scene representation.

1.2 Action-Conditioned Video Prediction

Details of Baselines. Previous works, including PhysGaussian [4] and PhysDreamer [5], used MPM operated on 3D Gaussians to simulate and render motions, but they do not consider robot-object interactions. Our baseline MPM uses the same simulation setting as these works but also adds support for two types of robot end-effectors: cylindrical pusher and gripper.

We represent the cylindrical pusher as a moving rigid body that collides with objects. The rigid shape is coupled with the soft body using frictional contact. We represent the gripper as a "sticky" sphere that fixes the particle's velocity to be the same as the gripper's speed whenever a particle is within the sphere.

We use the CMA-ES algorithm, a derivative-free optimization algorithm, to optimize physical parameters for both MPM and FleX baselines. The cost function is the mean 3D Chamfer Distance over future timesteps. For each object instance, we run optimization for 25 iterations.

1.3 Model-Based Planning with MPPI

The model-based control pipeline operates as follows: Given the state space $\mathcal{S}$ and the action space $\mathcal{A}$, we define a cost function that maps from $\mathcal{S} \times \mathcal{A}$ to $\mathbb{R}$. Starting from an initial state $S_0 \in \mathcal{S}$, we iteratively sample actions $\{a_i\}_{i=0}^{T-1}$ within the action space. The learned dynamics model predicts the outcomes of these actions, and the MPPI trajectory optimization algorithm identifies the action sequence $\{a_i\}$ that minimizes the cost function. In our experiments, the cost function comprises a task-related term measuring the distance to the target state, along with penalty terms for infeasible actions and collision avoidance.

We apply the MPPI trajectory optimization algorithm for model-based planning. Given the dynamics model $S_{t+1} = f(S_{0:t}, a_t)$, the cost function we minimize is:

$$\mathcal{J}(a_{0:T-1}) = \phi(S_T) + l(S_0, S_T), \tag{5}$$

where the task term $\phi(S_T)$ measures the distance from the current state to the target, and the penalty term $l(S_0, S_T)$ produces high costs for infeasible actions.

Task Term. For all tasks, the cost term is defined as the Chamfer Distance between the current state S_T and the target state S^*:

$$\phi(S_T) = \text{CD}(S_T, S^*). \tag{6}$$

Penalty Term. For all tasks, the penalty cost is defined as:

$$l(S_0, S_T) = \max_{s \in \mathcal{P}_T} \mathbb{1}\{s \notin \mathcal{W}\} + \max_{s_{\text{eef}}, s_{\text{obj}} \in \mathcal{P}_0} \mathbb{1}\{\|s_{\text{eef}} - s_{\text{obj}}\| < d_{\min}\}, \tag{7}$$

where $\mathcal{W}$ is the robot workspace; $\mathcal{P}_t$ is the particle set in state S_t; s_{eef} and s_{obj} represent end-effector and object particles, respectively. Thus, the penalty term penalizes actions that cause the object

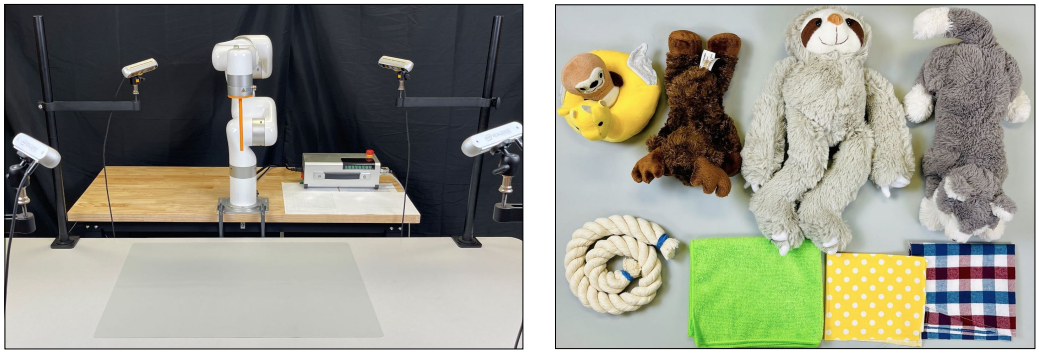


Figure 1: **Real-World Experiment Setup.** Left: our robot workspace with an xArm 6 robot and 4 fix-mounted RealSense D455 cameras. Right: the objects used in our experiments.

particles to move out of the workspace and actions that result in the end-effector contacting the object in S_0. We set $d_{\min} = 2$cm to avoid accidental collisions.

1.4 Real-World Experiments

Workspace Setup. In our real-world experiments, we utilize a UFACTORY xArm 6 robot with 6 DoF and a parallel gripper. For tasks such as toy animal relocating and rope straightening, we replace the original grippers with a 3D-printed cylindrical stick. For cloth relocating, we use the parallel gripper. Four calibrated RealSense D455 RGBD cameras are strategically positioned around the workspace, capturing RGBD images at 15Hz with a resolution of 1280×720. The robot manipulates objects within a 60 cm$\times$45 cm planar workspace, ensuring comprehensive data capture and precise manipulation. In Fig. 1, we show an overview of our workspace and the objects used in our experiments.

Real-World Data Collection. we teleoperate the xArm6 to interact with various objects using keyboard controls while maintaining a constant robot speed. For rope and toy animals, we attach a 3D-printed cylindrical stick to the end-effector and push the objects at different contact points. Specifically, we perform pushes of varying lengths, parallel to the workspace, covering a 2D action space from 3 cm to 15 cm. For cloth, we operate within a 3D action space, teleoperating the robot to grasp a corner of the cloth and move it from a start point to an end point using the gripper.

To ensure comprehensive data collection, we use four calibrated RealSense D455 RGBD cameras to capture RGBD images at 15Hz, synchronized with the robot actions. This setup allows us to collect multiview videos of the interactions, which are used to optimize tracking from 3D Gaussian Splatting and effectively train our dynamics models. We record 40-second episodes for rope and toy animals, and 30-second episodes for cloth to avoid severe deformations that are inefficient to model. This approach ensures high-quality data for model training.

2 Additional Results

We provide additional results showcasing our complete framework, including qualitative results in 3D tracking, action-conditioned video prediction, and model-based planning. Furthermore, we provide extended quantitative results of model-based planning on new objects. These comprehensive results and analysis demonstrate the effectiveness of our approach across various scenarios and object types.

2.1 Tracking with Dynamic 3D Gaussian

In Fig. 2, we present qualitative results from an additional view of rope and cloth, visualizing the tracking trajectories over 30 timesteps. The results demonstrate consistent tracking across different

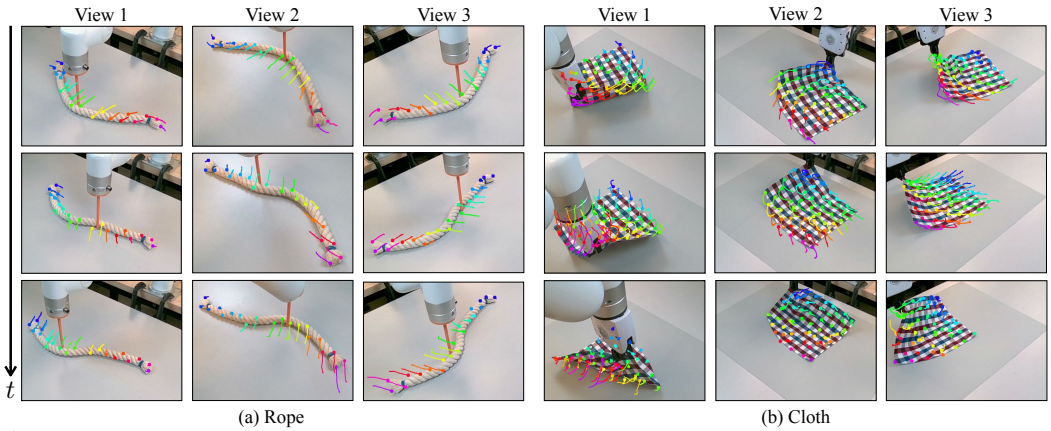


Figure 2: **Qualitative Results of 3D Gaussian Tracking Trajectories.** We demonstrate point-level correspondence on objects across various timesteps. The results also demonstrate consistent tracking performance across different views.

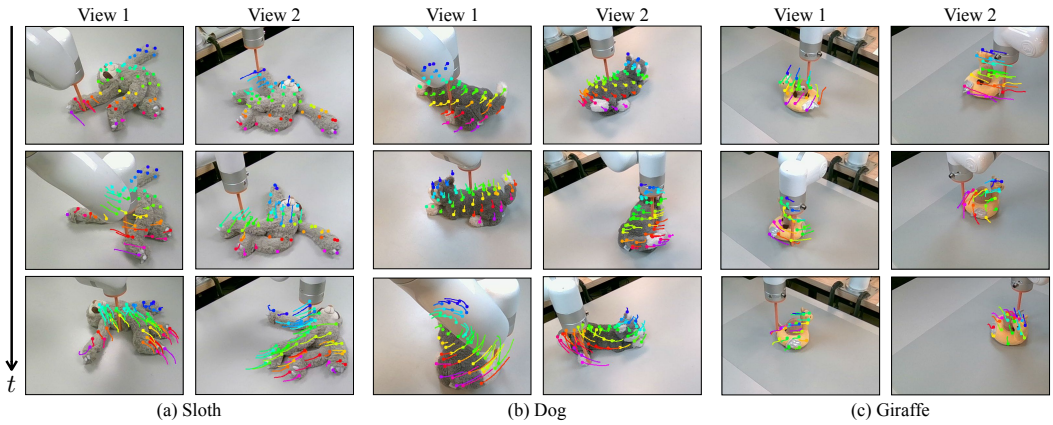


Figure 3: **Qualitative results of 3D Gaussian tracking trajectories on various toy animals.** 3D tracking results for toy animals with varying stiffness: (a) sloth (least stiff), (b) dog (moderately stiff), and (c) giraffe (most stiff). These results highlight our method's ability to maintain robust tracking performance despite differences in physical properties, effectively managing diverse scenarios and interactions.

views and precise tracking of various object parts. For example, when pushing the middle of the rope, the middle section moves with the cylindrical pusher, while the ends shrink inward or expand outward. Similarly, when the gripper grasps only the top of the cloth and moves, the bottom remains on the table with minimal movement, while the upper portion moves with the gripper. This showcases our method's ability to accurately capture and track object dynamics.

In Fig. 3, we present additional 3D tracking results for various toy animals, each with different physical stiffness properties. Specifically, the sloth (Fig.3 (a)) is the least stiff, the dog (Fig.3 (b)) is less stiff, and the giraffe (Fig. 3 (c)) is the stiffest. Our 3D tracking method demonstrates superior performance across all object instances, effectively handling diverse scenarios, interactions, and various robot actions and contact points. This showcases the capability of our approach to accommodate different physical properties and interaction dynamics by using various physical principles as optimization objectives for tracking.

2.2 Action-Conditioned Video Prediction

In Fig. 4, we present additional qualitative results of action-conditioned video prediction for 4 more instance episodes: 2 toy animals (deer and sloth), plaid cloth, and rope. Our method demonstrates high-quality alignment with the ground truth contours compared to the MPM baseline, indicating

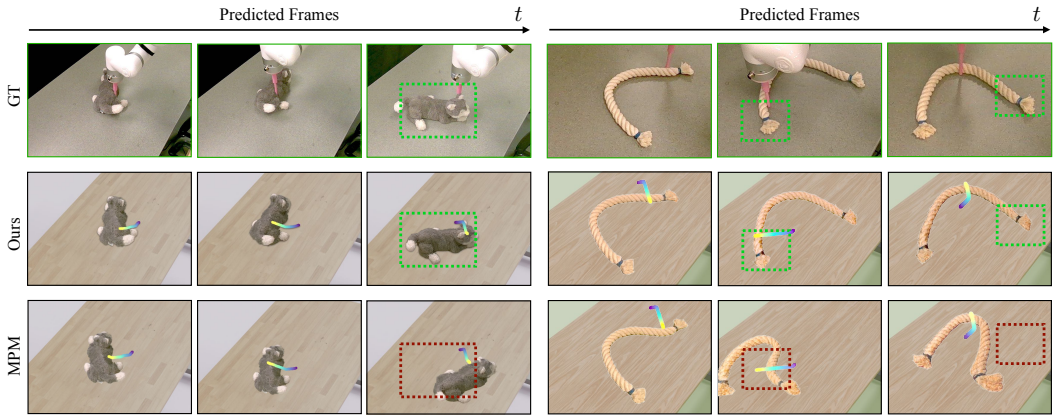


Figure 4: **Qualitative Results of Action-Conditioned 3D Video Prediction.** Our videos are generated by rendering predicted Gaussians on virtual backgrounds. Robot trajectories are visualized as curved lines, with yellow indicating current end-effector positions and purple representing historical positions. Compared to the MPM baseline, our video prediction results align more closely with the ground truth frames (GT), demonstrating superior accuracy. Our method achieves high-quality alignment with ground truth contours, significantly outperforming the MPM baseline in modeling object dynamics. This results in more realistic and physically accurate video predictions.

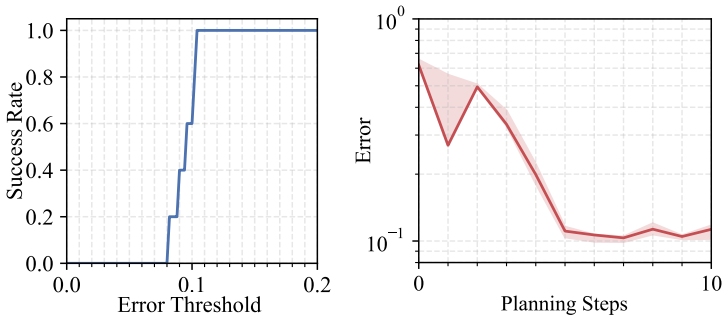


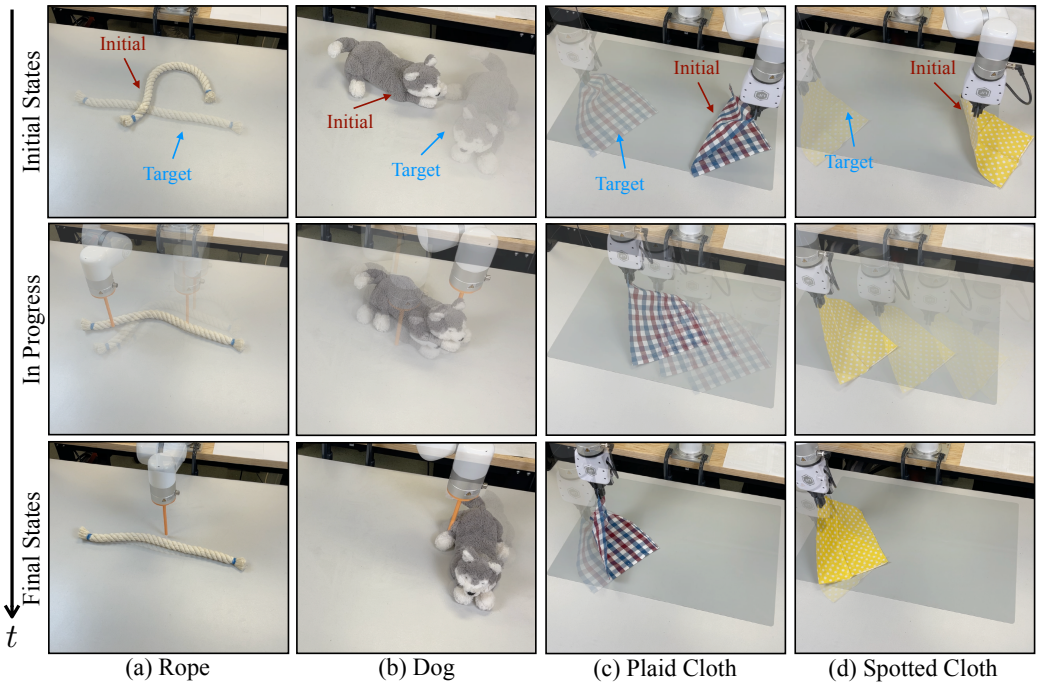
Figure 5: **Additional results of model-based planning.** For the cloth relocating task, we maintain the same basic configurations for planning perform each experiment 5 times and present the results as follows: (i) the median error curve relative to planning steps, with the area between 25 and 75 percentiles shaded, and (ii) the success rate curve relative to error thresholds.

a more accurate modeling of object dynamics. This accuracy enables more realistic and physically accurate video prediction. For example, over time, the MPM baseline increasingly deviates from the ground truth when manipulating the rope, and the behavior of the cloth fails to align with real-world physics. Our method maintains fidelity to the actual dynamics, ensuring more precise and reliable predictions.

2.3 Model-Based Planning

In Fig. 5, we present additional results of model-based planning on cloth. Our approach maintains low errors within a limited number of planning steps and achieves a high success rate under a stringent error margin. This demonstrates the effectiveness and precision of our method in handling cloth manipulation tasks.

In Fig. 6, the qualitative results of model-based planning on various objects illustrate that our framework accurately learns object dynamics, enabling efficient manipulation to target configurations within a few planning steps.



(a) Rope (b) Dog (c) Plaid Cloth (d) Spotted Cloth

Figure 6: **Additional real-world results of model-based planning.** We perform model-based planning tasks in the real world on various objects and tasks. By presenting the initial and target states, along with several in-progress steps, we demonstrate the efficiency of our learned dynamics in enabling effective model-based planning in robotics.

3 Computational Complexity of Planning

While MPM and FleX are theoretically capable motion models for planning tasks, using them as baselines for comparison is impractical due to their high computational complexity and inefficiency in real-world applications. To illustrate this, we compared the time required for an open-loop trajectory optimization. MPM requires 8.05 seconds for a single forward inference with a batch size of 10 (further increasing the batch size is not possible due to CUDA memory limitations). Our method takes only 0.61 seconds for the same forward inference with a batch size of 500, and just 0.16 seconds with a batch size of 10. Additionally, FleX takes 19.9 seconds with a batch size of 10. The results demonstrate that MPM and FleX are extremely time-consuming, making them unsuitable for practical planning tasks where efficiency is essential.

4 Ablation Study

4.1 Objectives for Dynamic 3D Gaussian Tracking

Configurations	Objectives					Metrics			
	$\mathcal{L}_{\text{rigid}}$	$\mathcal{L}_{\text{rot}}$	$\mathcal{L}_{\text{iso}}$	$\mathcal{L}_{\text{bg}}$	Consistent	3D MTE↓	3D δ_{avg} ↑	2D MTE↓	2D δ_{avg} ↑
Ours	✓	✓	✓	✓	✓	**6.90**	**89.26**	**4.92**	**93.27**
No $\mathcal{L}_{\text{Rigid}}$	×	✓	✓	✓	✓	18.32	59.19	15.48	64.30
No $\mathcal{L}_{\text{Rot}}$	✓	×	✓	✓	✓	8.36	84.41	7.92	86.26
No $\mathcal{L}_{\text{Iso}}$	✓	✓	×	✓	✓	10.89	79.96	9.14	82.30
No $\mathcal{L}_{\text{Bg}}$	✓	✓	✓	×	✓	32.94	47.82	36.27	53.18
No Consistent Prop	✓	✓	✓	✓	×	40.21	42.34	38.28	47.35

Table 1: Quantitative evaluation by systematically removing each objective one by one to observe its impact on the results. This analysis helps in understanding the contribution of each objective to the overall performance and effectiveness of the tracking method.

We evaluate the impact of removing each objective from our Dyn3DGS-based tracking method by systematically removing each objective one by one to observe its effect on the results. Additionally, we assess the impact of not maintaining uniform Gaussian attributes—constant number, color, opacity, and size—while allowing position and orientation changes. This analysis provides valuable insights into the significance of each objective and the importance of maintaining uniform Gaussian attributes for achieving optimal tracking performance. Specifically, we conduct this experiment on the rope category to provide evidence for analysis.

The quantitative results are shown in Tab. 1. These results demonstrate that physics-inspired regularization guides optimization effectively, ensuring physical plausibility and accurate long-term dense correspondence. By mirroring natural scene dynamics through non-rigid physical modeling principles, we enhance fidelity and stabilize tracking over time.

Among all the physical objectives, the local rigidity objective is critical, as its removal leads to a significant drop in tracking performance. Conversely, the rotation similarity objective has the least impact on tracking results. This objective is more related to the rendering quality of Gaussian Splatting, particularly affecting the scales of the Gaussian ellipsoids and the rotation effect on image quality. This is evident when we do not optimize the spherical harmonic (SH) coefficients in Gaussian Splatting [6], thereby reducing the parameters we need to optimize.

In our configuration, we use the background loss to ensure Gaussian points on the objects do not drift into the background. We focus solely on optimizing the Gaussians of the objects. Without the background loss, there is no explicit mechanism to force Gaussian points that have drifted into the background to return to the objects, significantly impacting tracking quality. This objective is crucial for maintaining accurate and reliable tracking.

Optimizing Gaussian attributes simultaneously with tracking can lead to confusion during the optimization process, particularly when occlusions occur, such as those caused by the robot. This results in a drop in tracking performance. By assuming consistent Gaussian properties, we make the tracking process more reliable, especially under occlusion scenarios, ensuring more accurate and stable tracking.

4.2 Deploying Graph-Based Dynamics Model on Gaussians

To bridge the gap between graph-based dynamics models which operates on sparse control points, we deploy Linear Blend Skinning (LBS) [7] for dynamics model inference. One important design choice in this process is to resample control points during test-time forward prediction, which ensures the uniformity of graph particles and reduces error accumulation. We thus evaluate the impact of resampling LBS control points on dynamics model prediction performance using three representative objects: plaid cloth (Cloth), deer (Toy) and rope (Rope).

We show our results in Tab. 2. We can observe that with resampling, the dynamics inference accuracy and video prediction accuracy are improved, highlighting the effectiveness of LBS in maintaining graph uniformity and enhancing Gaussian rendering quality. For ropes, the performance with and without resampling are comparable. One explanation is that resampling control points will not cause significant graph structure changes due to its linear shape. Overall, the perceptual similarity of predicted videos, measured by LPIPS, are consistently higher with resampled graph vertices, demonstrating that the technique is beneficial for more stable dense Gaussian motion interpolation with LBS.

5 Additional Limitation Discussions

5.1 Limitation of Tracking Performance

One limitation of the perception module is that the tracking would fail when there are significant textureless objects. The key to the problem is that tracking optimization is supervised by image rendering. For textureless objects with no significant patterns, these gradients from rendering super-

Metrics	Cloth w/ RS	Cloth w/o RS	Toy w/ RS	Toy w/o RS	Rope w/ RS	Rope w/o RS
3D Chamfer ↓	**0.051**	0.057	**0.027**	0.033	0.053	**0.050**
3D EMD ↓	**0.050**	0.057	**0.034**	0.037	**0.048**	0.049
$\mathcal{J}$ score ↑	**0.548**	0.524	**0.669**	0.651	0.419	**0.432**
$\mathcal{F}$ score ↑	**0.479**	0.451	**0.692**	0.680	0.667	**0.695**
$\mathcal{J}\&\mathcal{F}$ score ↑	**0.514**	0.488	**0.680**	0.665	0.543	**0.564**
LPIPS ↓	**0.024**	0.033	**0.019**	0.031	**0.025**	0.031

Table 2: Ablation results of resampling control points as graph vertices (abbreviation: RS). We present the dynamics model inference and video prediction results on three object instances.

vision fail to provide meaningful guidance for tracking optimization, making the tracking fail while maintaining reasonable rendering. As a result, the tracking can become inaccurate or unstable. This limitation suggests a potential area for future work, where alternative strategies, such as integrating additional sensory data or leveraging more robust features, could enhance the tracking accuracy. Moreover, the optimization of Dyn3DGS [1] requires the points to be tracked are visible at the initial state. To the best of our knowledge, no existing learning-based methods can track invisible points without priors of them. This is a known limitation across similar learning-based methods. In our framework, maintaining the object to be in a straightened state is a reasonable assumption, as learning object dynamics through the graph-based neural network requires constructing the typology of objects through graph representation. A promising potential for future work is to explore methods for obtaining geometric priors of objects, which enables more powerful tracking capabilities.

5.2 Training Time of Dyn3DGS Tracking.

We utilize Gaussian-based tracking as it provides dense correspondence and reconstruction of objects simultaneously, which is essential to the action-conditioned video prediction. However, the training of Dyn3DGS tracking takes hours, which limits its applications in more real-world scenarios. In the future, we aim to improve the efficiency of tracking optimization, such as proposing a more sparse representation upon Gaussian Splatting to reduce the particles to be optimized and enhance the framework's capabilities in more applications.

5.3 Generalization Capability of Dynamics Model

Our approach requires training a separate dynamics model for each object category, as the model cannot generalize across different categories. However, it can effectively generalize across different instances within a specific category. With just 10 minutes of interaction video, we can capture the unique physical properties of a category and generalize to unseen actions for that category. This level of training is sufficient to develop a robust dynamics model that accurately predicts action-conditioned video frames. Our results demonstrate that our method outperforms traditional approaches, such as MPM simulation and position-based dynamics (PBD) in FleX, in both accuracy and efficiency.

References

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

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

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

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

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

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

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