# Robotic Manipulation and Dense Tracking for Complex and Deformable Object Dynamics

Yunchao Yao

CMU-RI-TR-25-40

May 7, 2025


The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

**Thesis Committee:**
Professor Jeffrey Ichnowski
Professor Christopher G. Atkeson
Uksang Yoo

*Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.*

# Abstract

Many everyday human actions—such as adjusting the grip on a tool or folding clothes—require the ability to manage complex dynamics involving shifting contact, deformability, or high-speed motion. While these tasks are intuitive for humans, object manipulation under such conditions poses significant challenges for robotic systems. Enabling robots to perform complex dynamic manipulations would substantially expand their capabilities and real-world applicability.

This thesis investigates two complementary aspects to address these challenges from both the action and perception perspectives. The first part introduces SWIFT, a trial-and-error optimization method that enables a soft robotic hand to perform a challenging dynamic in-hand manipulation task—pen spinning—without relying on explicit dynamics modeling or simulation. This contribution demonstrates that dynamic, non-prehensile manipulation can be achieved using a surprisingly simple gradient-free optimization approach and highlights the promise of soft robotic systems in executing fast, contact-rich manipulation tasks.

The second part focuses on the perception side of manipulation and addresses the difficulty of accurately capturing the dynamic behavior of deformable objects. This is critical for downstream applications such as object-centric imitation learning or the creation of digital twins. To this end, the thesis presents DeformGS, a dense tracking method that leverages Gaussian splatting and a learned deformation model to accurately and densely track points on deformable objects from visual input.

Together, these two contributions advance the understanding of robotic manipulation in complex dynamic environments and suggest promising directions for the development of more capable, adaptive robotic systems.

# Acknowledgments

I would like to express my deepest gratitude to everyone who supported me throughout my Master of Science in Robotics journey. The completion of this thesis marks not only an academic milestone but also a deeply personal one. It would not have been possible without the guidance, encouragement, and generosity of many individuals.

First and foremost, I am profoundly grateful to my family and step-family. Their unwavering support — both emotional and financial — made this endeavor possible. Their belief in me, even during moments of uncertainty, gave me the strength to persevere and grow.

I am deeply grateful to my advisor, Professor Jeffrey Ichnowski, whose mentorship has been foundational to my development throughout this journey. His generosity with his time, his thoughtful guidance, and his unwavering support have made an immeasurable difference in my research and growth. Whether I was stuck on a technical challenge or navigating broader questions, he was always ready to offer insight, encouragement, and a clear path forward. More than just helping me solve technical problems, he actively taught me how to think and work like a researcher, from structuring projects and developing ideas to writing papers and presenting my work. Under his mentorship, I have grown significantly in both skill and perspective, and I am super grateful to have learned from him.

I would also like to sincerely thank Professor Christopher Atkeson, whose guidance has been both inspiring and transformative. He was always generous with his time and insights, offering perspectives that were deeply thought-provoking. Our conversations frequently challenged me to see problems from new angles and to approach research with greater depth and clarity. He also encouraged me to resist the urge to follow popular methods without careful reasoning and to remain intellectually honest and curious in my work. His influence has shaped the way I think about research, and I am truly fortunate to have the chance to learn from his mentorship.

Moreover, I would like to thank Uksang Yoo, who not only served on my committee but was also a key collaborator, lab mate, and friend. He played an essential role in the SWIFT project — from helping finalize its direction and setting up early infrastructure, to offering invaluable support

as deadlines approached. He generously dedicated his time during our time working together and was always ready to share thoughtful advice on both research and broader topics. I am truly grateful for his support and collaboration.

In addition, I would also like to thank Bardienus Duisterhof, who welcomed me as a collaborator even during the early stages of my master's studies. Working with him was an important learning experience, and I truly appreciated his openness to discussing research ideas and broader academic directions. I am grateful for the support and perspective he shared.

I would also like to extend a special thank to Jianren Wang, whose early discussions and exploration into robotic pen-spinning laid the foundation for what would eventually become the SWIFT project. Those initial conversations were more influential than they might have seemed at the time, and I remain grateful for that spark.

Last but not least, I would also like to thank my friends and colleagues — Hongyi Chen, Yuemin Mao, Shahram Najam Syed, Jeffrey Ke, Qiushi Zhang, Xilin Zhang, Arthur Jacobson, and members and alumni of the Momentum Robotics Lab — for their camaraderie, encouragement, and constant support. Their presence made this journey not only more manageable but also more enjoyable and fulfilling.

I feel deeply thankful to be part of the CMU community. The past two years have been filled with challenges, learning, and meaningful connections. This chapter of my life has shaped who I am, and I will carry its lessons and memories with me always.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Robotic manipulation in complex environments continues to pose significant challenges, especially for tasks that go beyond quasi-static interactions and rigid-body assumptions. Many everyday scenarios—such as folding clothes, preparing food, tossing objects, or flipping tools in hand—involve rapidly changing contact conditions, infinite-dimensional state spaces, or high-speed motion. These interactions often involve complex dynamics, making them challenging to model or control. Although such tasks are physically complex, humans perform them with ease and adaptability, highlighting the performance gap that remains in robotic manipulation. If robots could operate robustly and efficiently under these conditions, it would significantly broaden the scope of robotic capabilities in domains such as homes, healthcare, logistics, and manufacturing.

Recent advances in robotic manipulation have significantly expanded the range of tasks robots can perform in environments characterized by complex dynamics. Methods such as diffusion policy [16] or transformer-based policies [94] have enabled robots to capture diverse behavior across a wide range of scenarios. At the same time, improvements in teleoperation systems have made it easier to collect large volumes of high-quality demonstration data to train manipulation policies [27, 66]. However, collecting such demonstrations remains challenging for tasks that involve contact-rich interactions or dynamic motion. In these settings, human operators often struggle to respond to unpredictable object behavior in real time, making teleoperation-based demonstrations difficult to execute reliably, particularly for tasks

like in-hand manipulation, object flipping, or fast regrasping.

Reinforcement learning offers an alternative pathway and has shown promise in handling complex, dynamic manipulation through trial-and-error learning [35, 51]. However, the data requirements of reinforcement learning typically necessitate training in simulation, and this introduces significant challenges when transitioning to real-world deployment. In particular, the sim-to-real gap becomes pronounced in environments with deformable objects, soft robotics, or unstable contact dynamics, where accurately modeling the system is inherently difficult. These limitations highlight the need for methods that can operate directly in the real world while still handling the complexity of object behavior and uncertain interactions.

To tackle these challenges, particularly the ones in simulation and modeling for manipulation under complex dynamics, one way to approach the problem is to design systems that minimize reliance on these components. Another complementary direction is to directly improve the modeling and simulations to be more accurate and realistic. Motivated by these potential solutions, this thesis explores the two complementary perspectives toward manipulation under complex dynamics: the action and control perspective, and the perception and modeling perspective. These reflect two distinct but compatible aspects for dealing with the uncertainties that arise in real-world interaction.

From one direction, we ask: given limited realism in simulations or limited human demonstrations, how can a system discover suitable actions to perform dynamic, contact-rich tasks? In this thesis, we investigate how structured action spaces, compliant hardware, and optimizing directly with real-world trial-and-error can enable effective dynamic manipulation without relying on detailed models or simulators. From the other direction, we consider the challenge of understanding and representing complex physical interactions: If precise analytical modeling is challenging, can we instead build data-driven representations that capture the task-relevant aspects of the environment for downstream manipulations? This defines our perception and modeling perspective, where we develop approaches for constructing object-centric, visually derived models that support tracking and understanding of the dynamics of the environment, even in scenes with heavy deformation, occlusion, or visual ambiguity.

Together, these perspectives reflect a broader goal: to develop manipulation

systems that are physically adaptive and capable of operating in real-world settings. The two directions are explored through the two main contributions of this thesis, presented in Chapters 2 and 3.

The first contribution, presented in Chapter 2, is SWIFT—an optimization-based system that enables a soft robotic hand to perform dynamic in-hand manipulation using only real-world trial-and-error. A central feature of SWIFT is its reduced action space, which enables practical sampling in physical hardware. SWIFT demonstrates that it is possible to learn dynamic behaviors without relying on simulation, explicit models of object dynamics, or prior knowledge of physical properties. It also highlights the potential of soft robotic manipulators for fast, contact-rich manipulation tasks.

Chapter 3 introduces DeformGS, a novel method for dense 3D tracking of deformable objects in dynamic scenes using multi-view video. DeformGS builds on recent advances in Gaussian splatting and neural scene representations to recover fine-grained scene flow, even in the presence of shadows, occlusions, and large deformations. It further incorporates physics-inspired regularization to produce accurate and temporally coherent object trajectories. Although DeformGS is developed independently of manipulation tasks, its ability to produce accurate motion and deformation tracking opens possibilities for more informed and intelligent manipulation, object-centric imitation learning, and digital twin construction.

Finally, Chapter 4 concludes the thesis by summarizing the key insights from both contributions and discussing future research directions.

# Chapter 2

# Soft Dynamic pen spinning

## 2.1 Introduction

In-hand dexterity is crucial for many tasks common in our daily lives [55], and the ability to re-orient objects in the hand and re-grasp them is useful to perform these tasks efficiently and effectively [14, 71]. The compliance of soft robotic end-effector deformable fingers allows them to be robust to perturbations [4, 68] and enables them to interact safely with their environments [69, 89]. Previous works on soft robotic end-effector dexterity have largely focused on slow quasi-static tasks such as grasping and slowly reorienting objects [4, 34]. However, compliance makes it difficult to move the fingers both quickly and accurately. Such limitations underscore the gap between soft robotic end-effectors and human hands that can fully exploit the dynamics of objects to efficiently re-orient and re-grasp various tools and objects.

Pen spinning is a challenging dynamic task even for humans to master. As a case study of how to enable soft robots to perform fast dynamic tasks (Fig 2.1), it can suggest methods for tackling fast manipulation tasks with a soft manipulator. Approaching the problem of in-hand object re-orientation dynamically allows the robot to perform the task efficiently in one continuous action sequence, as demonstrated in other dynamic manipulation tasks [7, 30, 36, 37, 45, 93]. Previous works on exploiting object dynamics for in-hand reorientation of objects relied on knowing the object properties, such as its weight and world parameters [56]. However, in practice, we may not know such parameters a priori. For example, in the case of pen spinning,

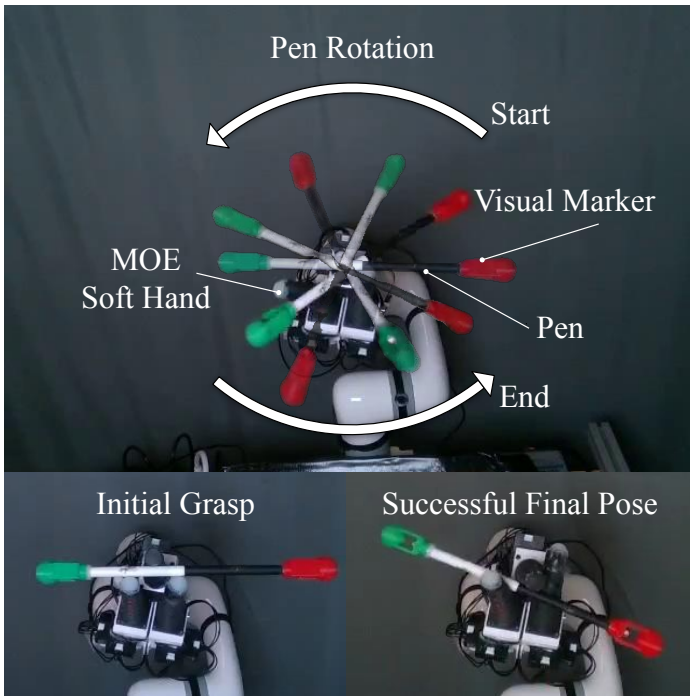



Figure 2.1: SWIFT tackles the problem of high-speed dynamic in-hand partially non-prehensile manipulation with soft robotic hands. Using a soft multi-finger gripper, the robot grasps a pen. Then, using a learned action sequence, rapidly rotates the pen around a finger and catches it.

visual observation may be inadequate to estimate the distribution of the weight and its center of mass to find the appropriate action parameters to successfully spin the pen. Moreover, the spinning motions usually last for less than a second, requiring high-speed sensing that is not always available and making close-loop control difficult or impractical. Rather than relying on knowing these parameters prior to interactions or using close-loop control, we propose Soft-hand With In-hand Fast re-orienTation (SWIFT), a system that learns where to grasp and how to dynamically manipulate objects autonomously through trial and error. A crucial component of SWIFT is the softness and compliance of the soft Multi-finger Omnidirectional End-effector (MOE) [89] enabling the system's ability to safely interact with the environment to pick up an object reliably and perform various dynamic manipulations.

The contact-rich nature of object re-orientation tasks, such as pen spinning, leads to a sizable gap between simulated environments and the real world, requiring extensive fine-tuning of simulators with real-world data [45, 78]. Additionally, the

soft robotic end-effector introduces unresolved challenges in realistically simulating contact interactions and complex soft-body mechanical phenomena, such as hysteresis and creep of soft material [49]. Therefore, we propose learning pen grasping and dynamic spinning skills with only real-world interactions. To this end, we define the task and desired behavior with an objective function evaluated with RGB-Depth camera feedback. We implement primitives for soft robotic pen spinning designed to reduce the dimension of the search space for a successful pen spin to 8 parameters. We use a soft and compliant robotic hand attached to a 6 degree-of-freedom robotic arm to repeatedly grasp and attempt to spin the pen. We deploy an evolution-based optimization system to efficiently explore the primitives' parameter space and narrow it down to a locally optimal set of parameters to successfully spin the pen. In experiments, we demonstrate that using SWIFT, a robot can learn to grasp and spin pens even if the properties of the pens, such as their weight or weight distribution, vary.

In summary, this chapter makes the following contributions:

1. Demonstrating a dynamic task of grasping and dynamically spinning a pen in a soft robotic hand,
2. Developing a self-supervised autonomous process to rapidly learn to spin a pen dynamically in the real world,
3. Evaluating our approach on a real robot under a variety of conditions.

## 2.2 Background

### 2.2.1 Related Works

Researchers have proposed various tasks and methods for dynamic manipulation of objects, such as throwing and catching of objects [5, 7, 48, 92], fast transport of grasped objects [37], and flinging rope or cloth [13, 17, 30, 87, 93]. In most of these previous works, the robotic arms provided an impulse to the object to move them dynamically [37]. Building on these foundations, this chapter explores a novel approach by leveraging a soft robot for in-hand dynamic reorientation.

### 2.2.2 Robotic Pen Spinning

Prior works have studied robot pen spinning [38, 56, 78]. Pen spinning is interesting because it involves many challenging aspects of in-hand dexterity, such as contact-rich interactions, partially non-prehensile manipulation, and object dynamics. Prior works have approached the task in one of two ways: analytical dynamics model-based control [56] and reinforcement learning aided by simulation environments [78]. Nakatani and Yamakawa [56] used a rigid robot hand, which is easier to model than the soft hand used in this chapter. Reinforcement learning-based approaches allow the researchers to define the task with reward functions that are hand-crafted [78] or semantically produced with language models [51]. Because of complex contact interactions and object dynamics of pen spinning, learning approaches have struggled with the sim-to-real transfer of policies trained in simulation and have only demonstrated quasi-static pen spinning with slow incremental reorientation. In contrast, our contribution in this chapter overcomes this sim-to-real gap by directly optimizing action parameters on real robots for dynamic pen spinning.

### 2.2.3 Soft Robotic Manipulation

Researchers have demonstrated the advantages of soft robotic manipulators in various quasi-static tasks, such as object grasping [61] and slow reorientation of regular objects, such as cubes [4]. Recent works have demonstrated methods to exploit soft robotic manipulators' inertial dynamics to accomplish tasks such as throwing efficiently [6, 31]. Similar to works in rigid dynamic manipulation, the focus has been on high-velocity control of soft robotic arms [10]. To our knowledge, our contribution in this chapter is the first to explore using soft robotic end-effectors for dynamic in-hand tasks such as pen spinning.

## 2.3 Problem Statement

The problem is to enable a soft robot hand to perform a high-speed in-hand rotation of an object. We focus on a specific instance of this problem: a pen spinning task similar to the "Thumbaround" trick performed by humans, where the pen is pushed

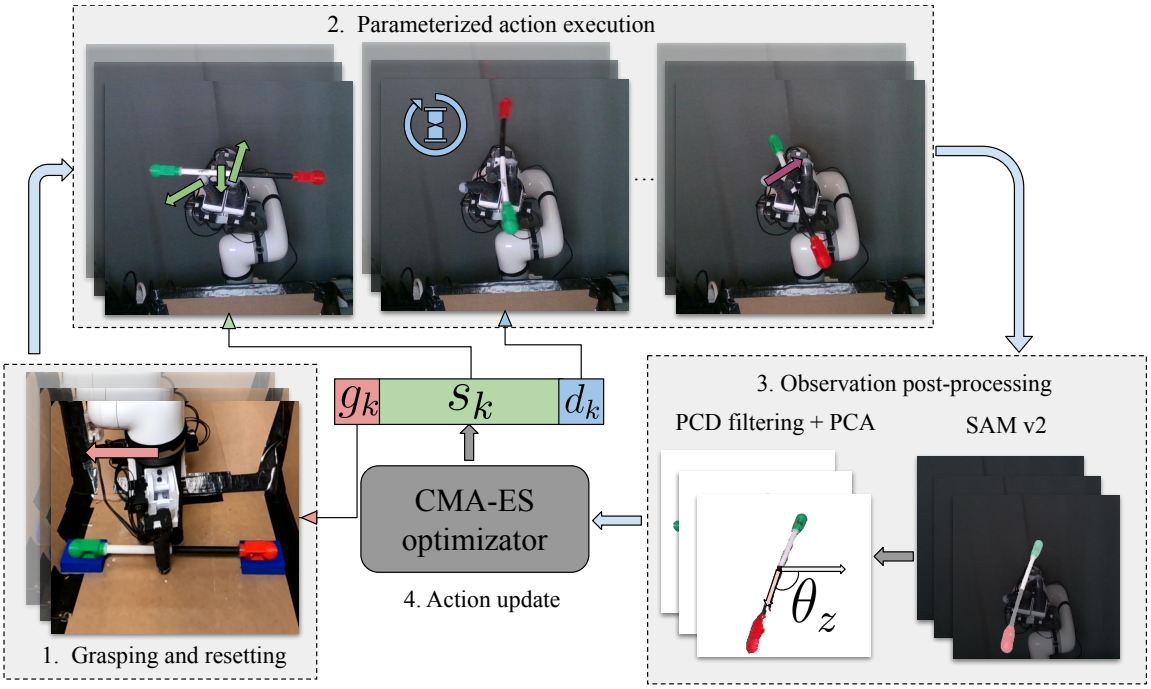



Figure 2.2: SWIFT optimization pipeline with four main stages. (1) The robot arm moves the MOE hand to a target grasp location. (2) It transitions to a pre-spin configuration, where the MOE fingers execute a parameterized action. (3) An RGB-D camera captures the trial, and SAM-v2 segmentation processes the point cloud to extract the pen's rotation and displacement. (4) The pipeline evaluates the observed states and updates action parameters using CMA-ES.

by the middle finger and spins around the thumb before being caught by the index finger. We assume the object is generally long and cylindrically symmetric with a well-defined major axis, and the mass and size are within the hand's grasping and manipulation capabilities. A manipulator arm orients the soft-robotic hand for repeatable grasps, but does not participate in the high-speed manipulation. We define success as a full rotation of the object without dropping.

## 2.4 Method

The SWIFT pipeline (Fig. 2.2) learns to spin a pen through trial and error. It optimizes low-dimensional action parameterization for a soft-robot manipulator by repeating a 4-stage pipeline that evaluates and learns from its results. Stage 1: During

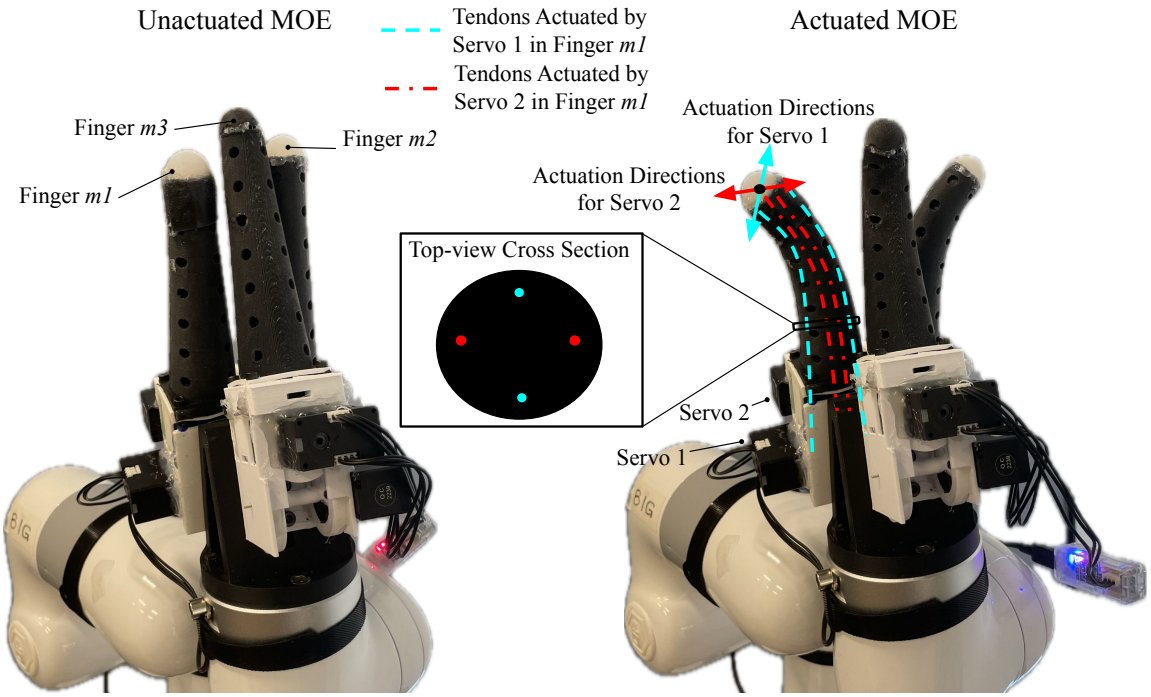



Figure 2.3: **Multi-finger Omnidirectional End-effector (MOE)**. The soft hand we used is a three-finger variant of the MOE. Each finger has four tendons actuated by two servo motors, with each servo controlling the finger in perpendicular directions.

grasping and resetting, the robot arm moves the MOE hand to a target grasp location. Stage 2: The robot arm then moves the MOE hand to the pre-spin configuration, where the MOE fingers execute the parameterized action. Stage 3: An RGB-D camera records the trial, and we apply post-processing on the point clouds to get the rotation and displacement state of the pen. Stage 4: The pipeline evaluates the objective function with observed states of the pen and updates the action parameters with the optimization algorithm CMA-ES.

### 2.4.1 Soft Hand

We design and build a low-cost soft-robot manipulator based on the *multi-finger omni-directional end-effector (MOE)* [90, 91] to perform the spinning task. The hand consists of three tendon-driven soft robot fingers. Each finger has two servos that pull tendons to bend the finger in perpendicular planes. Fig. 2.3 shows MOE in both

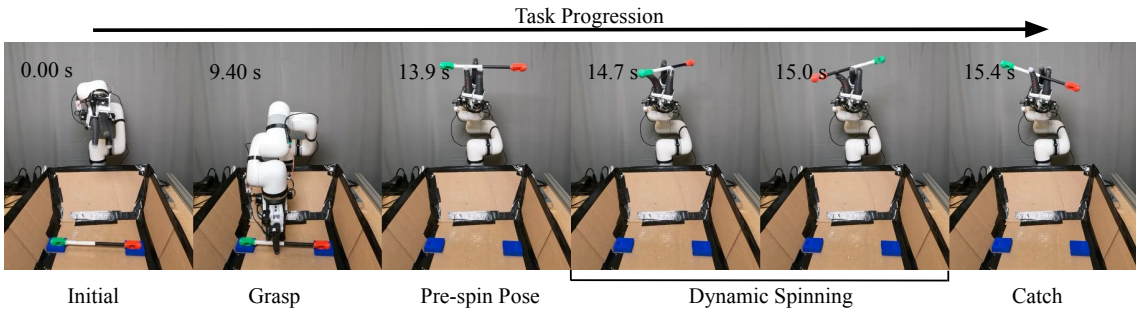



Figure 2.4: **Task progression over time**. There are three main stages for each pen-spinning trajectory. We place the pen in the blue slots fixed on the table. The robot moves to grasp at location $g$, and then moves the pen to the pre-spin pose. The MOE fingers then execute the spin action $s$, and finger $m1$ waits for $d$ seconds before closing to catch the pen. Finally, the robot arm moves to the initial joint configuration, dropping the pen and restarting the cycle.

actuated and unactuated states. Combining the servo motions can actuate each finger tip of MOE hand to reach locations on its hemispherical workspace. We attach two fingers (denoted $m1$ and $m2$) to one side of the hand base, and another finger ($m3$) to the opposing side. We attached the MOE hand to a 6-DOF robot arm.

### 2.4.2 Setup and Reset Procedure

To create a repeatable grasping process, before each attempt to spin the pen, we first manually place the pen in a fixture slot on the table (Fig. 2.4 Initial). The robot arm then executes a fixed set of movements to move the MOE hand to approximately the center of the pen. The MOE fingers then grasp the pen, and the robot arm carries the pen to a preset position and orientation before executing the spin action (Fig. 2.4 Grasp and Pre-spin Pose). We set the grasp pose so the robot hand picks up and holds the pen near its round fingertips. The MOE hand is slightly tilted to aid in catching after each spin. After manual tuning of the grasping and pre-spin configurations, this process reliably resets the system. We set up an RGB-Depth camera in front of the robot to capture trajectories. The camera's $z$-axis roughly points towards the $m3$ finger when the MOE hand reaches the pre-spin configuration. Fig. 2.5 shows the setup.

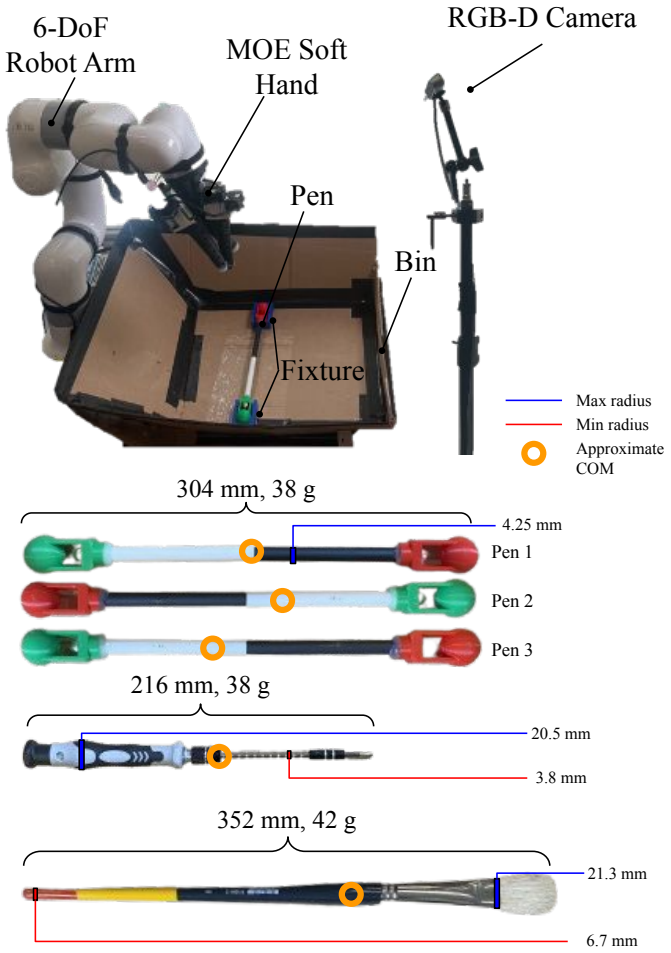



Figure 2.5: **Our setup for pen spinning.** Top: A 3-finger MOE soft robotic hand is attached to a 6 degree-of-freedom robot arm to develop a system that can safely interact with the pen and learn to spin it. An RGB-D camera is used to evaluate the performance of the sampled action based on the objective function. The box catches the pen when it is dropped to simplify resetting the system for the next trial. Bottom: the length, radius, weight, and approximate center of mass of each object used in the experiment

### 2.4.3 Pen Spinning Action Parameterization

We formulate the pen spinning task to be composed of a grasping action, a spinning action, and a catching action. To make learning more efficient, we parameterize the pen spinning actions into a reduced set of parameters consisting of servo targets, delay time, and grasping location. **Servo targets** are the target angles for each of

the servo's internal controllers to reach to spin the pen. With three fingers and two servos per finger, there are 6 servo target parameters. We denote this component as $s \in \mathbb{R}^6$. We choose to let $s$ represent servo angle changes with respect to the current servo angles. **Delay time** is the delay between the end of the spinning action and the start of the catching action. Inspired by human pen-spinning, we observed that only finger $m1$ needs to bend inward to catch the spinning pen, while fingers $m2$ and $m3$ can remain stationary. Therefore, during catching the servos on $m1$ simply reverse the angles used during spinning: if they executed $\theta_1, \theta_2$ during spinning, they will execute $-\theta_1, -\theta_2$ during catching. Since the pen's angular velocity varies depending on the spinning action, $m1$ must remain extended also for varying durations to avoid obstructing the spin. Thus, catching requires a single searchable parameter, $d \in \mathbb{R}$. **Grasping location** is a single parameter that controls the displacement of the grasping position relative to the center of the pen. We denote this parameter as $g \in \mathbb{R}$. The robot arm is pre-programmed to move the fingers to the center of the pen, and then adjust the end-effector position horizontally according to the grasping location parameter. The MOE fingers close according to a fixed sequence of motion to grasp the pen according to the grasping position. In evaluation, we denote the action parameterization that contains spinning action servo targets and delay time as $(s, d) \in \mathbb{R}^7$. We denote the action parameterization that contains all three components as $(s, d, g) \in \mathbb{R}^8$.

### 2.4.4 State estimation and Optimization objective

To evaluate the objective function for optimization of action parameters, the system observes the state of the pen using RGB images and a point cloud captured by the RGB-D camera at 30 fps. On the first frame of each trajectory, the system uses the Hough circle transform to locate red and green spherical markers on the pen. The system then uses the pixel coordinates of the centers of the spheres as initial key points for Segment Anything v2 [64] to create a dense segmentation mask on each frame of the pen along its trajectory. The segmentation masks help select 3D points belonging to the pen. Using a bounding box around the MOE fingertips, the system then removes outlier points that are outside of the bounding box. We consider the pen to be dropped in a frame if the filtered point cloud contains less than a threshold

number of points. To retrieve the rotation state of the pen that is robust against noisy RGB-D data, the system then applies PCA to the filtered point cloud, selecting the direction of the first principal component as the orientation vector of the pen. The system then computes the rotation angles of the pen in the camera's coordinate system.

The objective function contains a reward term and a fall penalty term. The system evaluates the objective function at each frame $t$ in a trajectory with $T$ total frames and sums the result. The rotation reward is

$$r_{\text{rot}} = \frac{\Sigma_{t=0}^{T} \mathbf{1}_{\|p_t\|>n} (\theta_z^t - \theta_z^{t-1})}{2\pi},$$

where $\theta_z^t$ is the rotation angle of the length of the pen around the $z$-axis in the camera coordinate frame at time step $t$. The indicator function $\mathbf{1}_{\|p_t\|>n}$ evaluates to 1 if the number of filtered points on the pen in a frame $p_t$ is greater than a threshold $n$; it is 0 otherwise. In all our experiments, we set $n = 500$. The depth camera points its $z$-axis towards and parallel to finger $m3$, and thus this rotation reward encourages rotation of the pen around the finger $m3$. The penalty term,

$$p_{\text{fall}} = 1 - \frac{\Sigma_{t=0}^{T} \mathbf{1}_{\|p_t\|>n}}{T},$$

penalizes frames where the pen is displaced too far away from the fingers according to the indicator function's threshold. We apply a weight factor $\lambda$ to combine both terms into the final objective function:

$$r = r_{\text{rot}} - \lambda p_{\text{fall}}. \tag{2.1}$$

### 2.4.5 Self-Supervised Primitive Parameter Optimization

SWIFT uses Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [32] to optimize the action parameters. CMA-ES is a gradient-free evolution strategy suitable for optimizing non-convex objective functions such as Eq. 2.1. At each generation, CMA-ES samples a population of action parameters from a multivariate normal distribution, parameterized by a mean and covariance matrix which are updated using the best performing candidates in the current generation. To prevent the robot arm

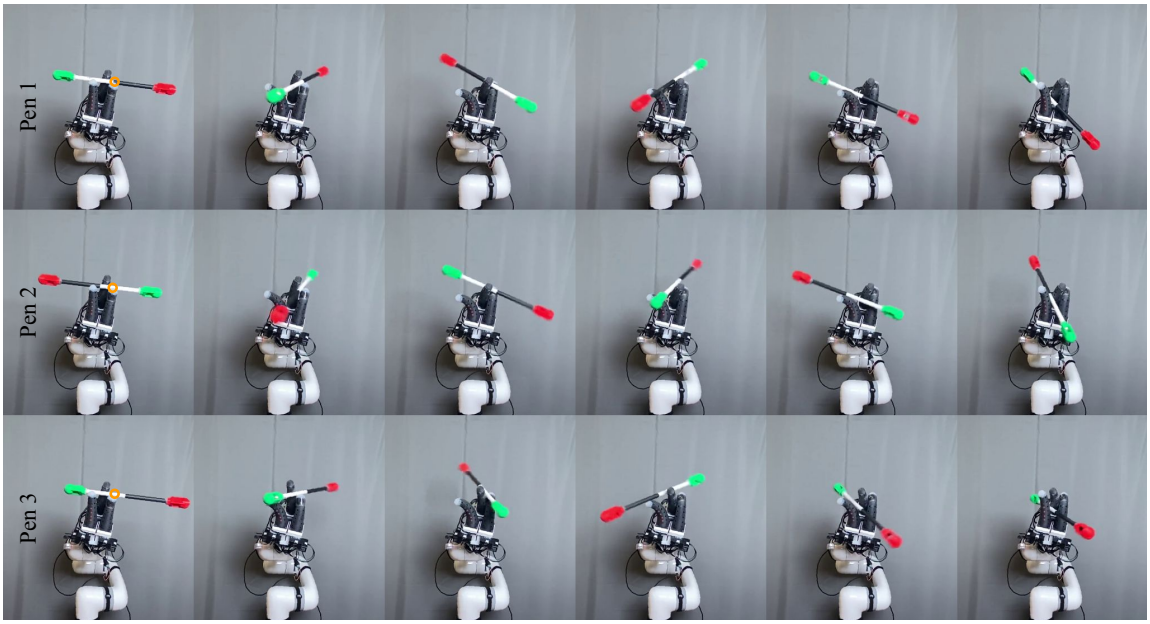



Figure 2.6: **Spinning visualization after optimization**. Top row: pen 1 with balanced weights. Middle row: pen 2 with unbalanced weight. Bottom row: pen 3 with unbalanced weight. The circle in the initial frame indicates the center of mass for the pen.

from moving to a grasping location off the pen and the MOE hand from executing actions beyond its mechanical constraints, we constrain the output of the optimization algorithm to always be within the allowable range. We conduct this optimization for each different object.

## 2.5 Evaluation

### 2.5.1 Experiment setup

We set up an environment (Fig. 2.5) for a repeatable pen grasp and camera observation. The setup also includes a bin to facilitate resets by a human. We evaluate SWIFT using three pen configurations with varying physical properties and test its adaptability on a paintbrush, and a screwdriver. We separately optimize for each object.

The 3 pens are 304 mm long with a radius of 4.25 mm and are visually identical. Pen 1 weighs 38 g and is balanced, with its center of mass at the geometric center.

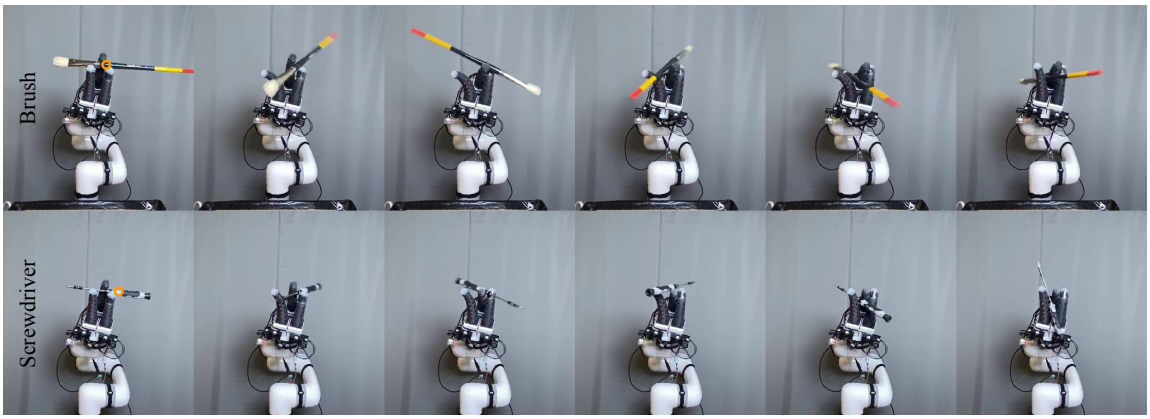



Figure 2.7: **Generalization to other objects**. We applied SWIFT to other objects with more irregular shapes, such as a brush or a screwdriver. The circle in the initial frame indicates the approximated center of masses.

Pen 2 weighs 26 g, with its center of mass offset by removing a detachable weight near the red spherical marker. Pen 3 also weighs 26 g, but has its center of mass shifted in the opposite direction of pen 2. The screwdriver weighs 38 g and has a length of 216 mm with a maximum radius of 20.5 mm and minimum radius of 3.8 mm. The brush weighs 42 g and has a length of 352 mm with a maximum thickness of 21.3 mm and minimum thickness of 6.7 mm.

We optimized the action parameters over 10 generations and evaluated the repeatability of the action parameters over 10 trials with the pen. Following the heuristics from the Hansen and Ostermeier [32], we choose the population size of CMA-ES to be $4 + 3 \log_2 8 \approx 13$ since our action parameterization has at most 8 dimensions. For evaluation on the brush and screwdriver, we optimize until the end of the first generation where we observe successful spins in the population or terminate at 10 generations. We chose the first manually observed success, which appeared after 4 generations, for evaluation instead of using the final CMA-ES value. Each experiment took roughly 2 hours to complete the generations and optimization.

### 2.5.2 Results

Fig. 2.6 shows successful pen spins after optimization. During optimization, the objective function we used only indirectly captures whether a spinning action is successful or not. A human observer labels trials a success or failure. A trail is a

Table 2.1: Action parameterization success rate We optimized various action parameterizations using 10 generations of SWIFT. The results suggest that optimizing both grasp location and spinning parameters yields the best performance, with generalization demonstrated on non-pen objects with varying mass distributions and geometries.

| Action Parameterization | Parameters | Object | Successes |
|---|---|---|---|
| Initialization | $\emptyset$ | pen 1 | 0 / 10 |
| | | pen 2 | 0 / 10 |
| | | pen 3 | 0 / 10 |
| No grasp optimization | $(s, d)$ | pen 1 | 0 / 10 |
| | | pen 2 | 7 / 10 |
| | | pen 3 | 0 / 10 |
| Optimal action from Pen 1 | $(s, d, g)$ | pen 1 | **10 / 10** |
| | | pen 2 | 0 / 10 |
| | | pen 3 | 7 / 10 |
| Full optimization (ours) | $(s, d, g)$ | pen 1 | **10 / 10** |
| | | pen 2 | **10 / 10** |
| | | pen 3 | **10 / 10** |
| | | brush | 10 / 10 |
| | | screwdriver | 5 / 10 |

success if the pen spins over the finger $m3$ and does not fall off the fingers. Table 2.1 reports the success rates of each baseline and ablated method. We initialized the CMA-ES optimization with heuristically hand-crafted action parameters. However, directly applying the fixed action initialization without optimization does not lead to any success in all three pen settings, each failing with 0/10 success rates (row 1 in the table). The result indicates to us that optimizing the actions for the MOE hand specifically is important for the success in these tasks.

We compared SWIFT optimization with all of the action parameters against different action parameterizations and report the results in Table 2.1. In the *Initialization* row, SWIFT does not optimize the grasp action $g$ and always grasps the center of the pen. In the *No grasp optimization*, we optimize $(s, d)$, but not the grasp point $g$, and the robot always grasps the pen center.

With this experiment, we see the efficacy of optimizing only $(s, d)$ is highly object-

dependent. With a central grasp, we could only succeed for pen 2 with a success rate of 7/10. A reason for grasping the center of the pen's length working for pen 2 could be that the optimal grasping point for pen 2 is the closest to the center of the pen as we can see in Fig. 2.6. These results highlight the importance of optimizing for the grasping point and the spinning action parameters for the system to work well for varying pen properties. In the *Optimal action from Pen 1* row, we show the results of optimizing action parameters for pen 1, then applying the action to pens 2 and 3. This shows the necessity to update action parameters for each new object. Directly using the optimal $(s, d, g)$ parameters from pen 1 to pen 3 results in 7/10 successes, while the same set of parameters results in 0/10 successes on pen 2. We can see in Fig. 2.6 that the optimal grasping points between pen 1 and pen 3 are both left of the pen's center in the image frame. This may explain why the optimal actions from pen 1 had some success on pen 3. In these experiments, all three pens are visually identical and therefore depend on SWIFT's ability to interact with the object to search for optimal action parameters. In the *Full optimization (ours)* row, we found that optimizing $(s, d, g)$ for each object results in 10/10 success rates for all pens. The higher success rate for pen 2 using full optimization compared to not optimizing grasping also suggests that having the ability to search over the grasping position enabled the search for a more robust spinning motion.

Lastly, we experiment with SWIFT applied to two other objects: a brush and a screwdriver. Fig 2.7 shows the results of these generalization experiments. SWIFT achieves 10/10 and 5/10 success rates for the brush and screwdriver, respectively. The screwdriver is particularly challenging to spin due to its irregular shape. It is also noteworthy that the fixture structure for resetting was tailor-made for the pen rather than the brush or screwdriver, leading to greater variance in grasping, which can reduce the repeatability of a successful spin. However, SWIFT optimized the action parameters and achieved successful spins, highlighting SWIFT's versatility.

## 2.6 Limitations and future work

In this chapter, we present SWIFT, a robust system for dynamic in-hand pen spinning with a soft robotic end-effector. SWIFT leverages real-world interactions to learn from trial-and-error to optimize the pen grasping and spinning actions for soft robotic

end-effectors. Importantly, it does not require explicit knowledge of the object's physical properties, allowing the system to robustly spin pens even when the weight distributions and shapes are varied. By using a sampling-based optimization strategy, we were able to efficiently explore the action space and discover the optimal set of actions for pen spinning.

We demonstrated the system's robustness across pens of different weights and mass distributions, suggesting the ability to generalize and adapt to changes in object inertial properties that are not easily observable without interaction. Additionally, we tested SWIFT on two other objects, a brush and a screwdriver, and found that SWIFT still succeeded in optimizing action parameters to spin them. The results highlighted the effectiveness of soft robotic end-effectors performing dynamic manipulation tasks with contact-rich and dynamic conditions.

In the future, we aim to extend and generalize our approach to a broader range of objects beyond pen-shaped ones and explore additional in-hand dynamic tasks using soft robotic end-effectors. Future research can also focus on developing more generalizable policies that apply across multiple objects without requiring individual optimization, as well as systems that minimize heuristics.

To further improve the efficiency and robustness of dynamic manipulation systems, one direction may be to explore incorporating additional state estimation methods or using more sensory feedback, such as contact inference or proprioception. These could provide richer insight into the interaction between the manipulator and the object, potentially enabling more generalizable and robust control strategies. In the following chapter, we introduce a dense tracking framework that may support such estimation of complex dynamics for more adaptive manipulation systems.

# Chapter 3

# DeformGS: Scene Flow in Highly Deformable Scenes for Deformable Object Manipulation

## 3.1 Introduction

While the previous chapter demonstrated that complex, dynamic manipulation can be achieved through trial-and-error optimization without relying on explicit modeling or feedback, access to richer state information could further improve adaptability and generalization. In particular, estimating object deformation, contact, or other internal states may help inform future manipulation strategies that move beyond open-loop execution. In this chapter, we shift focus to the perception side of this challenge and explore how dense 3D tracking of deformable objects can provide structured representations that support such understanding.

Recent advances in robot learning have demonstrated impressive performance on challenging tasks, including rigid and deformable object manipulation. Scaling these approaches to deployment will require an improvement in robustness and learning from few demonstrations. A promising avenue for improving in robot learning performance are intermediate representations and foundation models, including 6D object pose estimation [19, 20, 46, 72, 80, 82, 86], semantic latent features [59], and 2D pixel-wise

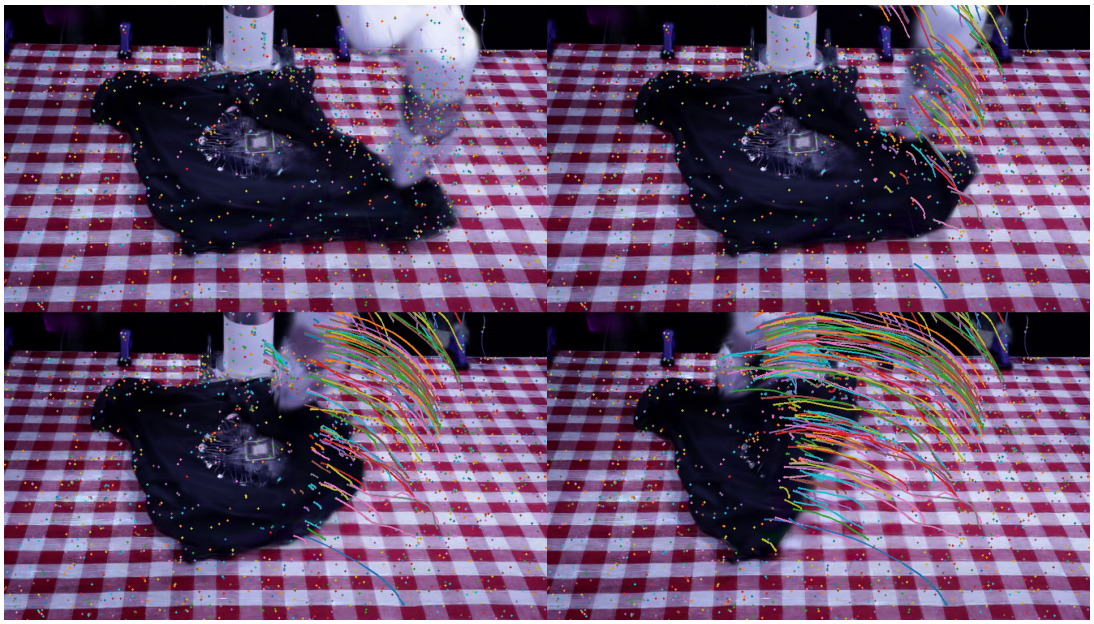


Figure 3.1: We propose DeformGS, a method that improves state-of-the-art methods for accurate 3D point tracking in highly deformable scenes. This figure shows DeformGS tracking in a real-world scene from the Robo360 [44] dataset.

tracking [39, 83]. However, perception and representations that will lead to robust manipulation of deformable objects remains an open challenge, due to self-occlusions, shadows, and varying (or lack of) textures.

Three-dimensional dense point tracking, or *3D scene flow*, is a useful representation for robot manipulation, as it provides flexibility to represent high-dimensional dynamic state changes, while the deformable objects drops, deforms, and drapes during manipulation. In particular, dense 3D scene flow can be an input to imitation learning policies [3, 83], can be used to learn a transition model [67], to identify and track task-relevant key points, or to create a digital twin through real2sim transfer. Recent work in monocular tracking has seen improvements in performance on datasets such as TAP-Vid [22], but it remains unclear how to effectively lift from 2D tracking to 3D for robotic spatial understanding in challenging highly deformable scenes.

To overcome these limitations, Gaussian Splatting provides a promising avenue. Recent work demonstrated Gaussian Splatting [40, 41] can yield state-of-the-art novel-view synthesis and rendering speeds exceeding 100 fps. Concretely, 3D Gaussian Splatting uses a fast differentiable renderer to fit the colors, positions, and covariances

of a set of 3D Gaussians. An extension of 3D Gaussian splatting [50] showed dynamic scenes can be modeled by explicitly optimizing the properties of Gaussians over time, resulting in novel-view synthesis and scene flow.

Explicitly optimizing the Gaussian properties as in Dynamic 3D Gaussians [50] may result in degraded performance with large deformations and shadows. The Gaussian properties may converge to local optima, especially in scenes with large deformations, strong shadows, and occlusions.

We propose DeformGS, a method that uses time-synchronized image frames from a calibrated multi-camera setup to track 3D geometries of deformable objects as they move through shadows and occlusions. DeformGS learns the canonical state of a set of Gaussians and a deformation function that maps the Gaussians into world space. This enables tracking by recovering scene flow, and novel-view rendering (through splatting) using a fast differentiable rasterizer.

We evaluate DeformGS in six photo-realistic synthetic scenes of varying difficulty. The scenes contain large deformations, shadows, and occlusions (Figure 3.1 shows tracking trajectories computed by DeformGS). Empirical results show that DeformGS infers 55.8 % more accurate 3D tracking results compared to previous state-of-the-art [50, 84]. In a scene with a 1.5 m $\times$ 1.5 m cloth (i.e., Scene 1), DeformGS can track cloth deformation with as low as 3.3 mm median tracking error.

We also evaluate DeformGS in the real world on the Robo360 [44] dataset. We show qualitative results for tracking rigid and deformable objects in cluttered scenes, and study two robotics applications: (1) real2sim transfer to create a digital twin, and (2) tracking task-relevant keypoints for downstream grasping applications.

In summary, our contributions are as follows:

- We provide the first approach to accurately perform 3D dense tracking of deformable objects using 4D Gaussians.
- We provide experiments that suggest state-of-the-art performance in 3D metric tracking of deformable cloth. DeformGS improves tracking accuracy by an average of 55.8% in synthetic experiments and demonstrates robust 3D tracking in the real world for deformable objects. The latter can be exploited as a representation for imitation learning and represents a new method for building digital twins.

- A set of six synthetic scenes with large deformations, strong shadows, and occlusions. The scenes and source code is available at the project website: https://deformgs.github.io

## 3.2 Related Work

### 3.2.1 Neural Rendering for Novel View Synthesis

DeformGS builds on prior work in novel-view synthesis, and uses photometric consistency as a signal to achieve 3D tracking. A popular novel view synthesis approach is NeRF [52]. It uses neural networks to learn scene representations that are capable of photo-realistic novel view reconstruction. Particle-based methods use a more explicit representation than typical NeRF-based approaches. DeformGS builds on 3D Gaussian Splatting [40, 41] which belongs to the latter category. 3D Gaussian Splatting [40] proposed a differential rasterizer to render a large number of Gaussian 'splats,' each with their state including color, position, and covariance matrix. Contrary to the NeRF-based approaches, Gaussian splatting achieves real-time rendering of novel views with state-of-the-art performance.

### 3.2.2 Dynamic Novel View Synthesis

The assumption of static scenes in neural rendering approaches prevents application to real-world scenarios with moving objects or humans, such as the dynamic and deformable scenes in this chapter. One line of work to address this assumption is adding a time dimension to NeRF modeling [24, 28, 43, 85]. Prior works either condition the neural field on explicit time input or a time embedding. Another line of work learns a deformation field to map 4D points into a canonical space [60, 63], i.e., every 4D point in space and time maps to a 3D point in a canonical NeRF. DeVRF [47] proposed to model the 3D canonical space and 4D deformation field of a dynamic, non-rigid scene with explicit and discrete voxel-based representations.

Several recent works extend the above approaches to 3D Gaussian splatting. Dynamic 3D Gaussians [50] explicitly models the position and covariance matrix of each Gaussian at each time step. This method struggles in dynamic scenes with

large deformations, strong shadows, or occlusions. We build on another recent work, 4D Gaussian splatting [84], which uses feature encoding techniques proposed in HexPlanes [11] and K-planes [26], and learns a deformation field instead.

### 3.2.3 Point Tracking

Point tracking methods, usually trained on large amounts of data, aided previous 3D tracking approaches by providing a strong prior [47]. We also construct several baselines that include point tracking methods (Section 3.6). Prior work on point tracking often studies tracking 2D points across video frames, where a dominant approach is training models on large-scale synthetic datasets containing ground-truth point trajectories [21, 23, 33, 95] or dense optical flows [79]. Optical flow [2, 73] or scene flow [1, 29, 75, 76] can also be viewed as single-step point-tracking in 2D and 3D, respectively.

Another relevant line of work tightly couples dynamic scene reconstruction and motion estimation of non-rigid objects. A predominant setup is fusing RGBD frames from videos of dynamic scenes or objects [57]. Tracking or correspondence-matching methods see a progression from template-based tracking of objects with known shape or kinematics priors (such as human hand, face or body poses) [12, 58, 65], to more general shapes or scenes [8, 9, 96]. The main difference from these works is that we do not use depth input, and perform more rigorous quantitative evaluations on tracking specific points.

Most related to ours are the more recent methods that obtain tracking from neural scene rendering. DCT-NeRF [77] learns a coordinate-based neural scene representation that outputs continuous 3D trajectories across the whole input sequence. PREF [70] optimizes a dynamic space-time neural field with self-supervised motion prediction loss. Most recently, Luiten et al. [50] models dynamic 3D Gaussians explicitly across timestamps to achieve tracking. While our contribution in this chapter also leverages 3D Gaussians, in contrast to the explicit modeling in Dynamic 3D Gaussians [50], we learn a deformation function that scales much better with video length, and we focus on deformable objects that are more challenging than the ball-throwing videos used in [50].

### 3.2.4 Tracking for Robotics

A core motivation for studying point tracking is the potential it can unlock for robotics applications: for example, RoboTAP [74] shows pre-trained point-tracking models improve sample efficiency of visual imitation learning. It detects task-relevant keypoints, infers where those points should move to, and computes an action that moves them there. Any-point [83] learns to predict keypoint tracks, but conditioned on language inputs. Track2Act [3] builds on Any-point by learning a generalizable zero-shot policy, which only needs a few embodiment-specific demonstrations.

Rigid-body, or 6D, pose tracking and estimation has a rich history in robotics due to it foundational ability to model the world for a robot to manipulate [19, 20, 46, 53, 72, 80, 81, 82, 86]. In this chapter, we propose a deformable object analog of 6D pose tracking with the aim of extending successes to deformable object manipulation.

While existing methods leverage 2D tracking, and learn an additional policy to output robot actions, DeformGS provides a more powerful representation that allows for reasoning directly in 3D, instead of in the 2D image space.

## 3.3 Problem Statement

Given a set of timed image sequences captured from multiple cameras with known intrinsics and extrinsics, the objective is to learn a model that performs 3D tracking and novel view synthesis. Each image sequence is captured over the same time interval $t \in [0, H]$.

**3D Tracking** The primary goal is to recover the trajectory of any point in a dynamic scene by modeling the deformation of Gaussians over time. Thus, the objective is to find a function $x_t = Q(x_0, t_0, t)$, where $x_0 \in \mathbb{R}^3$ is the location of a point of interest at a chosen time $t_0 \in [0, H]$, while $x_t \in \mathbb{R}^3$ is the location of the same point at another chosen time $t \in [0, H]$. The function $Q$ is valid for any point $x_0$ and any $t \in [0, H]$, allowing for tracking of any point in space.

**Novel View Synthesis** The secondary goal is to achieve accurate scene flow by using photometric consistency as a supervision signal. To achieve this, the objective is to recover novel views from arbitrary viewpoints. The extrinsics at any viewpoint can be captured by matrix $P$, with $P = K[R|T]$. Here $K$ is the intrinsics matrix,

$R$ is the rotation matrix of a camera with respect to the world frame, and $T$ is the translation vector with respect to the world frame. Concretely, the goal is to learn a function $V$ such that $I_{P,t} = V(P, t)$, where $I_{P,t}$ is an image rendered from a camera with extrinsics $P$ at time $t$. As with the tracking objective, the time parameter is valid for any $t \in [0, H]$.

## 3.4 Preliminary

### 3.4.1 Gaussian Splatting

3D Gaussian Splatting [40] deploys an explicit scene representation by rendering a large set of 3D Gaussians each defined by their mean position $\mu$ and covariance matrix $\Sigma$. Given $x \in \mathbb{R}^3$, its Gaussian is

$$G(x) = e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)},$$

Directly optimizing the covariance matrix $\Sigma$ would lead to infeasible covariance matrices, as they must be positive semi-definite to have a physical meaning. Instead, Gaussian Splatting [40] proposes to decompose $\Sigma$ into a rotation $R$ and scale $S$ for each Gaussian:

$$\Sigma = RSS^TR^T,$$

and optimize $R$, $S$, and the mean position.

Given the transformation W of a camera, the covariance matrix can be projected into image space as

$$\Sigma' = JW\Sigma W^TJ^T,$$

where $J$ is the Jacobian of the affine approximation of the projective transformation.

During rendering, we compute the color $C$ of a pixel by blending $N$ ordered Gaussians overlapping the pixel :

$$C = \sum_{i \in N} c_i \alpha_i \prod_{j=1}^{i-1}(1 - \alpha_j).$$

where $c_i$ is the color of each Gaussian and $\alpha_i$ is given by evaluating a Gaussian

with covariance multiplied with a learned per-Gaussian opacity $\sigma$ [40, 88]. This representation allows for fast rendering of novel views, and aims to reconstruct the geometry of the scene.

### 3.4.2 Deformation Fields for Dynamic Scenes

Prior work showed that a deformation function combined with a static NeRF in a canonical space can enable novel view synthesis in dynamic scenes. The deformation function $F_{\text{NeRF}} : \mathbb{R}^3 \to \mathbb{R}^3$ deforms a point in world coordinates $(x', y', z')$ into a point in canonical space $(x, y, z)$.

Prior work formulated $F_{\text{NeRF}}$ as an MLP [62] and a multi-resolution voxel grid [25]. Wu et al. [84] applied a similar approach to arrive at Gaussian splatting of dynamic scenes. Given the state of a single canonical Gaussian, defined by $P = [\mu, S, R, \sigma, C]$ at time $t$, a deformation function is

$$P' = F_{\text{4DGS}}(P, t),$$

where $F_{\text{4DGS}}$, similar to the Hexplanes [11], contains a neural-voxel encoding in space and time. 4D-GS [84] starts with a *coarse* stage for initializing the canonical space, by setting $P' = P$, bypassing the deformation field and learning canonical properties directly. During the *fine* stage we learn the deformation function.

We propose DeformGS (Figure 3.2), based on 4D-Gaussians [84], to render novel views in dynamic scenes. The key differences with 4D-GS are: (1) we propose an intuitive method to track canonical Gaussians in world coordinates using a continuous deformation function, (2) the output of the deformation function is different, e.g., DeformGS infers shadows and does not alter opacity or scale over time, (3) using the method shown in Figure 3.3, we enforce physics-inspired regularization losses on the 3D trajectories of Gaussians, and (4) DeformGS uses dynamic object masks as input to learn a per-Gaussian mask.

4D Gaussian Splatting

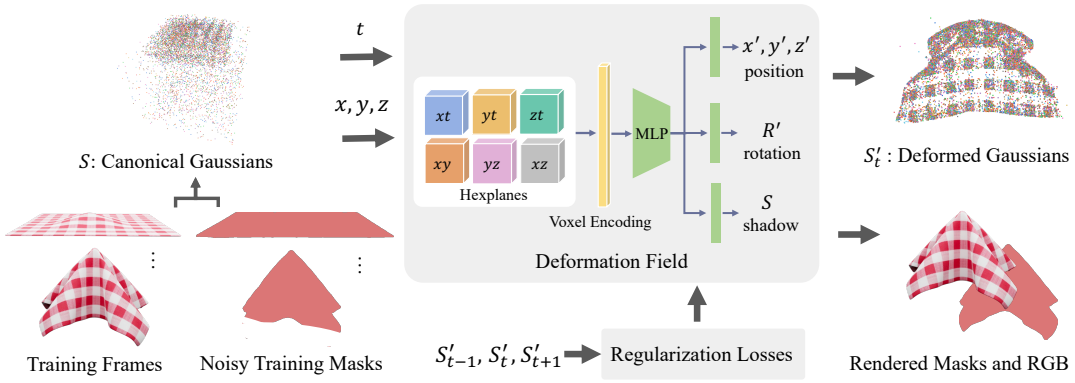



Figure 3.2: DeformGS maps a set of Gausians with canonical properties to world space using a deformation function $F$. The deformation function takes in the position of a Gaussian $(x, y, z)$ and a queried timestamp $t$, to infer shadow $s$, rotation $R'$ and metric position $(x', y', z')$. During training, we use the metric positions and rotations to regularize the deformation function, considering the state at $t = \{i-1, i, i+1\}$ with Gaussian metric states $P'_{t-1}, P'_t, P'_{t+1}$

## 3.5 Method

DeformGS achieves novel-view synthesis and high-quality 3D tracking using a canonical space of Gaussians and a deformation function to deform them to world space (Section 3.4.2). To incentivize learning physically plausible deformations, DeformGS introduces several regularization terms (Section 3.5.1). Finally, DeformGS learns 3D masks to focus regularization and Gaussian deformation on dynamic parts of the scene (Section 3.5.2).

**Canonical Neural Voxel Encoding.** As with prior work, DeformGS learns a deformation function $F$ from a canonical space. We use a neural-voxel encoding to ensure $F$ has sufficient capacity to capture complex deformations. Prior work [25, 26, 54, 84] showed that neural-voxel encodings improve the speed and accuracy of novel-view synthesis in dynamic scenes. We leverage HexPlanes [11, 84] to increase capacity for simultaneous 3D tracking and novel-view synthesis.

Figure 3.2 shows an overview of the canonical neural-voxel encoding. Each of the six voxel modules can be defined as $R(i, j) \in \mathbb{R}^{h \times lN_i \times lN_j}$. Here $\{i, j\} \in \{(x, y), (x, z), (y, z), (x, t), (y, t), (z, t)\}$, i.e., we adopt HexPlanes in all possible combinations. $h$ is the size of each feature vector in the voxel, $N_i, N_j$ are the sizes of the HexPlanes in each dimension, $l$ is the upsampling scale. In every module, each

plane has a different upsampling scale $l$. To query the multi-resolution voxel grids, we query each plane using bilinear interpolation to finally arrive at a feature vector used by the deformation MLP.

**Deformation MLP.** The deformation MLP takes in the voxel encoding and uses the encoding to deform the canonical Gaussians into world coordinates. Figure 3.2 shows the deformation MLP, which infers position, rotation, and a shadow scalar, given a feature vector from the neural voxel encoding. We choose this set of outputs to model rigid-body transformations of each Gaussian and changes in illumination. Modeling changes in illumination is critical in the presence of shadows. We multiply the RGB color of each Gaussian by the shadow scalar $s \in [0, 1]$, and the shadow scalar is in the range $[0, 1]$ by feeding the output of the MLP through a sigmoid activation function.

Next, we deform the Gaussians, modifying their mean positions $\mu$ and rotation $R$, and arrive at a set of Gaussians in the world space each with state $P'$. The differentiable rasterizer from Gaussian Splatting [40] then renders the Gaussians to retrieve gradients for regressing both the canonical Gaussian states and the parameters of the deformation function.

Unlike 4D-Gaussians [84], we propose to not infer opacity or scale using the deformation field. Optimizing for opacity and scale over time would allow Gaussians to disappear or appear instead of following the motion, which would make tracking less accurate. This design choice reduces the capacity of the deformation function, hence a lower view reconstruction quality as compared to 4D-Gaussians is expected.

### 3.5.1 3D Tracking using 4D Gaussians

**Physics-Inspired Losses.** Figure 3.3 shows the process of tracking Gaussians from the canonical space in world space. By querying the deformation function $F$, we can track the position of a Gaussian along the entire trajectory.

Without additional supervision, this approach will not necessarily converge to physically plausible deformations. Especially when objects include areas with little texture and uniform color, the solution space for all deformations is underconstrained by photometric consistency alone. To learn a more grounded deformation function, we propose regularization terms inspired by physics.

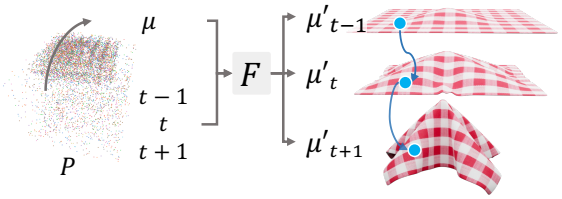



Figure 3.3: DeformGS uses three adjacent timesteps at every iteration to enforce physics-inspired regularization terms. All Gaussians are deformed to world space using the deformation function $F$, and rasterized to compute the photometric loss and its gradients. We use the positions of the Gaussians in world coordinates to compute the regularization terms based on local isometry and conservation of momentum (Section 3.5.1).

After empirically evaluating several combinations of regularization terms, we adopt the isometry loss proposed in [50] and add a conservation of momentum term. The first term captures a local isometry loss, which we compute based on the state of the $k$ nearest neighboring (KNN) Gaussians.

**Local Isometry Loss** We incentivize the Gaussians to keep the relative position of the $k$ nearest neighbors constant w.r.t. $t = 0$. With sufficient deformation, this assumption will be broken at a larger scale, but at a local scale, this regularization avoids drift from the ground-truth trajectory. The isometry loss is

$$\mathcal{L}_t^{\text{iso}} = \frac{1}{k|\mathcal{P}|} \sum_{i \in \mathcal{P}} \sum_{j \in \text{knn}_i} w_{i,j} \left| \|\mu_{j,0} - \mu_{i,0}\|_2 - \|\mu_{j,t} - \mu_{i,t}\|_2 \right| .$$

with

$$w_{i,j} = \exp\left(-\lambda_w \|\mu_{j,0} - \mu_{i,0}\|_2^2\right) ,$$

Here $\mathcal{P}$ is the set of all Gaussians.

**Conservation of Momentum** We add a term to incentivize conservation of momentum. Newton's first law states objects without external forces applied, given some mass $m$ and velocity vector $\mathbf{v}$, maintain their momentum $m \cdot \mathbf{v}$. We introduce the regularization term

$$\mathcal{L}_{i,t}^{\text{momentum}} = \|\mu_{i,t+1} + \mu_{i,t-1} - 2\mu_{i,t}\|_1.$$

This term incentivizes a constant-velocity vector and has the effect of imposing a

low-pass filter on the 3D trajectories. It smooths out trajectories with many sudden changes of direction and magnitude (momentum).

### 3.5.2 Learning 3D Masks

Learning accurate 3D tracking in scenes with a mix of static and dynamic objects and rich textures poses significant challenges, mainly: (1) imposing physics-inspired regularization terms on all Gaussians may cause issues when dynamic and static objects interact, and (2) modeling millions of dynamic Gaussians can become a significant computational burden.

To address this, DeformGS takes noisy masks of dynamic scene components such as cloth, and learns what Gaussians are dynamic. More formally, we render a mask $M$ by

$$M = \sum_{i \in N} m_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j).$$

where $m_i$ is a per-Gaussian property, with $m_i \in [0, 1]$. We then add a regularization term to the loss function s.t. $m_i$ is regressed to best reconstruct M. Finally, DeformGS uses $m_i$ to select a subset of Gaussians to be dynamic, and applies regularization terms only to those Gaussians.

## 3.6 Experiments

We evaluate DeformGS on synthetic and real-world datasets of scenes with highly deformable objects. Section 3.6.1 provides details on the simulation experiment setup, evaluation metrics, and baseline methods. Section 3.6.2 reports evaluation results from the compared methods and provides analysis. Section 3.6.3 lists the real-world evaluation setup, and finally in Section 3.6.4 we provide a qualitative evaluation of the performance of DeformGS in the real-world.

### 3.6.1 Simulation Experiment Setup

**Dataset Preparation** We use Blender to model dynamic cloth sequences and render photo-realistic images. We create 6 distinct scenes, each containing a different cloth

with distinct visual and physical properties, and we render images from 100 different camera views and 40 consecutive time steps for training for a total of 4,000 images. The cloth deformations are introduced by dropping each cloth over one or a few invisible balls onto a ground plane or by constraining the cloth at an attachment point. We obtain ground-truth trajectories by tracking the mesh vertices of deformable objects in Blender. Every scene contains a single deformable object and a rendered background.

**Oracle Baselines** We compare DeformGS to 2D tracking oracle models which have access to ground truth depth and trajectory information. While these methods were not designed for 3D tracking, they are well-known for their impressive 2D tracking performance. Their numbers aid in putting the tracking performance of the other baselines into context. We run RAFT [73] on all views, project tracking to 3D using ground truth depth, and report the mean results as the RAFT model. We provide two additional oracle methods which have access to the ground-truth trajectories as well. *RAFT Oracle* first evaluates on all views, to then output only the trajectories from the view with the lowest median trajectory error (MTE). We also report *OmniMotion Oracle*, which runs OmniMotion [79] on the viewpoint with the lowest MTE for RAFT. Training OmniMotion takes roughly 12–13 hours per viewpiont on an NVIDIA RTX 4090 GPU, making training on all 100 views impractical. The numbers from *RAFT Oracle* and *OmniMotion Oracle* are not an apples-to-apples comparison with the other methods, as to obtain their result they had to access privileged ground-truth trajectories.

**Gaussian Splatting Baselines** (1) Dynamic 3D Gaussians (*DynaGS*) [50], which also builds on 3D Gaussian splatting for dynamic novel-view synthesis, except it explicitly models the positions and rotations of each Gaussian at each time-step. This results in straightforward tracking of any point via finding the trajectory of the learned Gaussian closest to a queried point. Although the original paper assumes a known point cloud at the first frame, we instead use a randomly sampled point cloud for a fair comparison, with DynaGS and DeformGS both not using depth information.

(2) Finally, we compare to tracking using 4D-Gaussians [84] (*4D-GS*). We add the approach for 3D tracking of canonical Gaussian, as shown in Figure 3.3, to extract 3D trajectories from a novel view synthesis model. Comparing to *4D-GS* serves to show the impact of the changes made in the model architecture, the regularization

terms, and using learning per-Gaussian masks to arrive at DeformGS.

**Training and Evaluation Setup** We create a dataset of 6 dynamic cloth scenes, each with varying physical and visual properties (Figure 3.1). For DeformGS and 4D Gaussians, we perform 30,000 training iterations, and set point cloud pruning interval to 100, voxel plane resolution to [64, 64], and multi-resolution upsampling to levels $L = \{1, 2, 4, 8\}$. We set the regularization hyper parameters (Section 3.5.1) for all synthetic scenes to $\lambda_w = 2{,}000$, $\lambda^{\text{momentum}} = 0.03$, $\lambda^{\text{iso}} = 0.3$, and $k = 20$ for KNN. We keep all hyper parameters the same for real-world scenes, but increase the regularization terms for momentum and isometry loss. We generate the masks with segment anything (SAM) [42] for the initial frame, and use XMem [15] to propagate to future frames.

For DynaGS, we set $\lambda^{\text{rigid}} = 4$, $\lambda_w = 2{,}000$,$\lambda^{\text{iso}} = 2.0$, and $k = 20$, as in the open-source code.

We evaluate each compared method on 1,000 randomly sampled points on each cloth.

### 3.6.2 Simulation Results

**3D Point Tracking** Following prior work [50, 95], we report median trajectory error (MTE), position accuracy ($\delta$), and the survival rate with a threshold of 0.5 [m] [50].

The results are summarized in Table 3.1. We make the following observations: (1) DeformGS outperforms baselines RAFT, DynaGS, and 4D-GS, by achieving a MTE of 55.8% - 76.0% lower compared to the baselines. (2) The discrepancy between the RAFT oracle model and its averaged result demonstrates the difficulty arising from frequent self-occlusions. (3) The oracle models perform very well, this is in part thanks to the falling and short-horizon nature of these sequences, limiting self-occlusions. In the real-world we expect much larger errors due to noisy depth and more challenging occlusions in long-horizon tasks. It would also be unclear what viewpoint to choose without access to ground truth trajectories. This also points to future research avenues for additional supervision through optic flow and 2D tracking algorithms such as RAFT. (4) Scenes with less texture such as scene 3 perform significantly worse than scenes with strong texture.

**Qualitative Results** Figure 3.4 shows DeformGS results on all synthetic Blender

| Metric | Method | Scene 1 | Scene 2 | Scene 3 | Scene 4 | Scene 5 | Scene 6 | Mean |
|---|---|---|---|---|---|---|---|---|
| 3D MTE [mm] ↓ | RAFT [a] | 67.264 | 89.944 | 220.125 | 177.909 | 84.593 | 23.422 | 110.543 |
| | RAFT Oracle[ab] | 3.381 | 26.956 | 58.971 | 12.481 | 3.930 | 3.192 | 18.152 |
| | OmniMotion Oracle [ab] | 0.535 | 14.513 | 39.958 | 4.556 | 2.487 | 2.011 | 10.677 |
| | DynaGS | 24.233 | 81.119 | 464.64 | 54.074 | 34.985 | 36.101 | 115.859 |
| | 4D-GS | 4.645 | 95.032 | 223.839 | 27.441 | 14.091 | 11.619 | 62.778 |
| | **DeformGS** | **3.373** | **45.33** | **88.369** | **14.022** | **7.257** | **8.173** | **27.754** |
| 3D $\delta_{\text{avg}}$ ↑ | RAFT [a] | 0.553 | 0.379 | 0.222 | 0.533 | 0.586 | 0.619 | 0.482 |
| | RAFT Oracle[ab] | 0.926 | 0.577 | 0.411 | 0.703 | 0.833 | 0.808 | 0.710 |
| | OmniMotion Oracle [ab] | 0.987 | 0.693 | 0.549 | 0.864 | 0.885 | 0.849 | 0.805 |
| | DynaGS | 0.624 | 0.31 | 0.042 | 0.435 | 0.533 | 0.527 | 0.4118 |
| | 4D-GS | 0.902 | 0.339 | 0.164 | 0.583 | 0.697 | 0.715 | 0.5667 |
| | **DeformGS** | **0.929** | **0.522** | **0.322** | **0.71** | **0.856** | **0.787** | **0.688** |
| 3D Survival ↑ | RAFT [a] | 0.945 | 0.792 | 0.779 | 0.822 | 0.792 | 0.854 | 0.831 |
| | RAFT Oracle[ab] | 0.986 | 0.833 | 0.957 | 0.872 | 0.929 | 0.903 | 0.913 |
| | OmniMotion Oracle[ab] | 1 | 0.963 | 1 | 0.985 | 0.963 | 0.933 | 0.977 |
| | DynaGS | 0.999 | 0.99 | 0.483 | 0.988 | 0.992 | 0.992 | 0.907 |
| | 4D-GS | **1** | 0.967 | 0.834 | **1** | 0.999 | **1** | 0.967 |
| | **DeformGS** | **1** | **1** | **1** | **1** | **1** | **1** | **1** |

[a] Method had access to ground truth depth. [b] Method had access to ground truth trajectories to pick the best camera view.

Table 3.1: 3D tracking results on the deformable cloth dataset (Figure 3.1). For each metric, the methods above the solid line had access to privileged information, see [ab] and Section 3.6.1 for more details. The results suggest that DeformGS outperforms the baselines in all averaged metrics, and is competitive with the oracle models. The results also suggest our novel deformation function architecture, learning per-Gaussian masks, and physics-inspired regularization losses improve the tracking performance compared to 4D-GS [84]. We do not consider the oracle methods to be fair baselines and therefore do not bold their results.

scenes, Figure 3.5 shows ground truth and inferred trajectories in scene 5. The results show that especially DynaGS and 4D-GS introduce large errors as the cloth drapes down. RAFT improves over DynaGS and 4D-GS but requires accurate depth estimation.

### 3.6.3 Real-World Experiment Setup

**Robo360 Data** The Robo360 dataset [44] is a 3D omnispective multi-material robotic manipulation dataset. It covers many different scenario's, including manipulation by robot manipulators and humans captured by 86 calibrated cameras. These properties make it an ideal dataset to evaluate the effectiveness of DeformGS in the real world. We select three scenes: (1) a human folding a larger duvet, (2) a human folding a

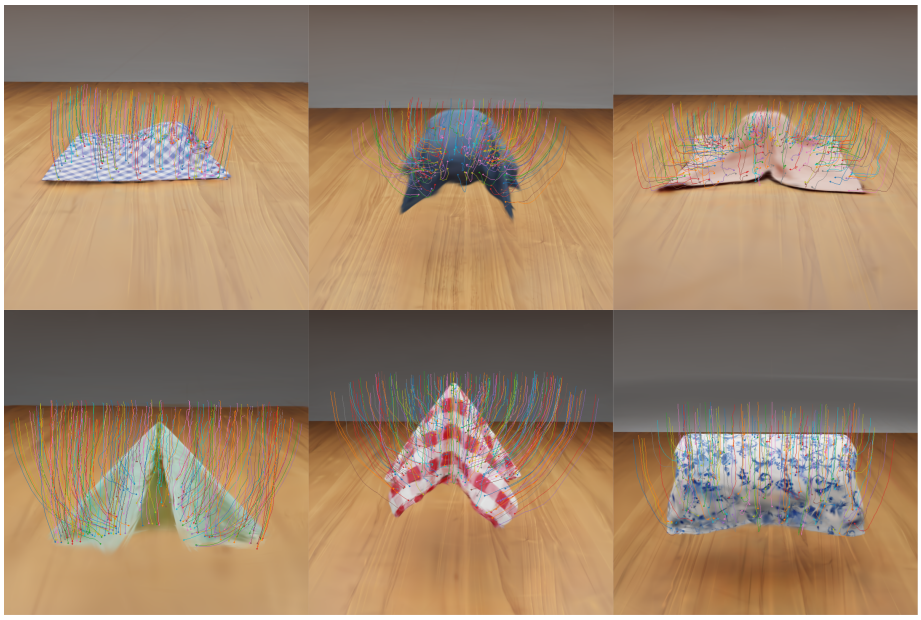
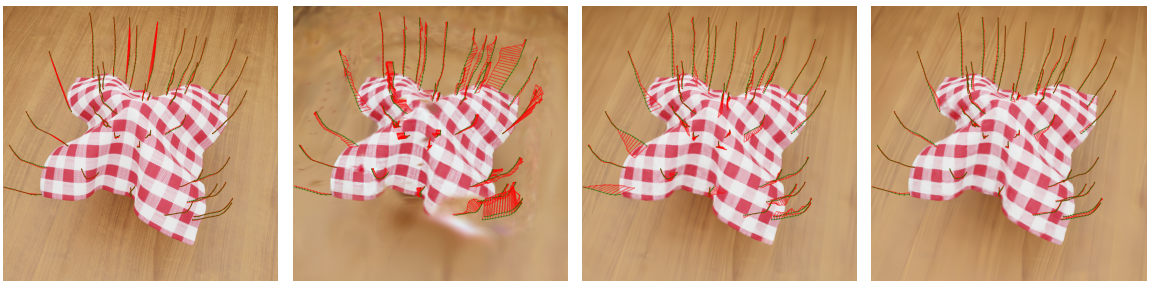


Figure 3.4: This figure shows the rendering and tracking of DeformGS in the six dynamic Blender [18] scenes used for evaluation. We will refer to the scenes in this Figure as Scenes 1, 2, 3, 4, 5 and 6 ordered from left to right, and top to bottom.



RAFT DynaGS 4D-GS DeformGS

Figure 3.5: **Results on Scene 5**: randomly sampled ground-truth trajectories in green, inferred trajectories in red, and the error of corresponding points in red lines. Compared to the baseline methods, DeformGS results in smaller errors in 3D tracking.

smaller cloth, and (3) a robot arm folding a t-shirt. In the cloth folding scene, we exclude viewpoints where the entire person's body is visible to eliminate unnecessary complexity.

We also subsample the data to demonstrate DeformGS performance with fewer views. The duvet folding scene contains 17 training views, the cloth folding scene contains 20 training views, and the t-shirt folding robot scened was trained with 21

views.

### 3.6.4 Real-World Experiment Results

**Real2Sim for Digital Twins** Figure 3.6 shows the 3D tracking overlaid on rendered images, as well as the Gaussian points at each time step. The results suggest that DeformGS is able to successfully infer smooth and meaningful trajectories in the real world. While no ground truth is available, the trajectories appear to follow their geometry closely except for a few floating Gaussians. Hyperparameter tuning of the regularization functions, as well as discarding Gaussians with a low opacity, might help resolve this.

The point cloud included in this Figure can be used to create a digital twin after recording the sequence. The digital twin of the duvet, and the entire environment, can then be used to create more dense supervision for imitation learning approaches.

**Task-Relevant Keypoint Tracking** Robotic manipulators can benefit from tracking task-relevant keypoints, such as the corner of a cloth or the edge of a jacket. Figure 3.7 shows a comparison between 4D-GS [84] and DeformGS in 3D point tracking, evaluated on both duvet and cloth scenes. The results suggest DeformGS leads to more smooth and overall useful trajectories. The trajectories from 4D-GS intertwine into more messy trajectories, and appear less physically plausible. This would hinder the adoption of 3D tracking into robot applications. Figure 3.1 shows DeformGS trajectories for a robot folding a t-shirt.

## 3.7 Conclusions

In this chapter, we address the challenging problem of 3D point-tracking in dynamic scenes with deformable objects. We introduced DeformGS, the first approach that learns continuous deformations for 3D tracking of highly deformable scenes. We empirically demonstrate that DeformGS outperforms baseline methods and achieves both high-quality dynamic scene reconstruction and high-accuracy 3D tracking on highly deformed cloth objects with occlusions and shadows, both in simulation and the real world. We also contribute a dataset of six synthetic scenes to facilitate future research.

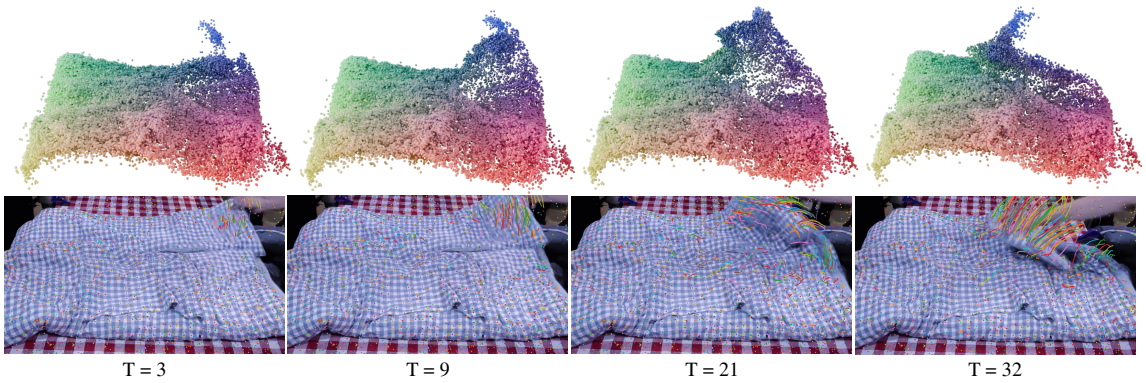



Figure 3.6: A person manipulating a duvet in the Robo360 [44] dataset, reconstructed using DeformGS. The top row shows the 4D Gaussians as point clouds, where the color represents dense correspondences. The bottom row shows rendered views overlaid with 3D trajectories projected to image space.

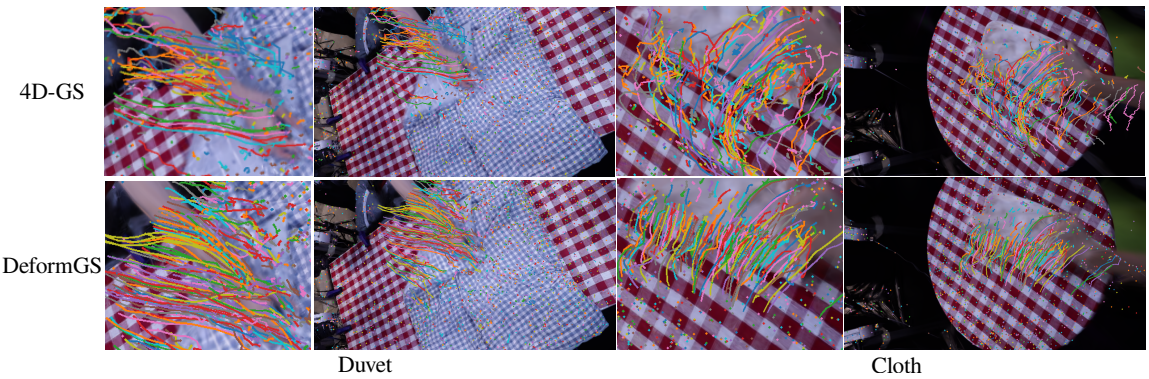



Figure 3.7: Real-world results comparing our proposed DeformGS against 4D-GS [84]. The 3D trajectories inferred by DeformGS appear more smooth and accurate, whereas 4D-GS displays more cluttered trajectories.

**Limitations and Future Work** DeformGS, similar to prior work on dynamic novel view reconstruction, requires a setup of multiple synchronized and calibrated cameras, which may require a significant engineering effort in real-world scenarios. Additionally, significant innovation will be required to achieve the demonstrated results in real-time, as will be beneficial for scalable robot applications.

While DeformGS improves upon prior methods, we do observe Gaussians wandering off in some cases. We also notice the algorithm is relatively sensitive to the regularization hyperparameters ($\lambda^{\text{momentum}}$ and $\lambda^{\text{iso}}$), this might be resolved in the future by adding supervision from state-of-the-art point-tracking algorithms. These

limitations point to promising directions for future research.

# Chapter 4

# Conclusion

This thesis explored two complementary perspectives on robotic manipulation for environments with complex dynamics. The first contribution explored the action and control side through the development of SWIFT, a system that enables a soft robotic hand to perform dynamic in-hand pen spinning using black-box optimization. By structuring the action space appropriately and exploiting the mechanical compliance of the soft robot manipulator, SWIFT demonstrates how challenging dynamic manipulation can be accomplished without relying on explicit models or simulation.

The second contribution approached the problem from the perception and modeling perspective, introducing DeformGS, a dense 3D tracking method for deformable objects in dynamic scenes. DeformGS combines Gaussian splatting with learned Gaussian deformation models to recover dense and physically plausible object tracking from visual input, even under heavy self-occlusion, shadowing, and large deformations. This dense tracking also enables a variety of downstream applications in robotics, such as object-centric imitation learning and the creation of digital twins for deformable objects.

While these two contributions were developed independently, they offer mutually reinforcing lessons. SWIFT shows that even model-free action can be robust if the action space is well-designed and can effectively exploit the physical structure of the hardware. DeformGS, on the other hand, shows that task-relevant state estimation can be recovered from vision, even under challenging conditions, without requiring complex ground-truth modeling.

Combining the insights of the two contributions, a potential future direction lies in applying DeformGS to track the deformation of the soft robotic hand during dynamic manipulation. By capturing dense representations of the hand's deformation, this tracking could reflect complex system states, such as contact behavior or robot-object interactions. Such a representation may serve as a structured model of the system's physical dynamics, supporting the learning of more reactive, closed-loop policies. This offers a potential alternative to the black-box, open-loop optimization strategy used in SWIFT.

More broadly, this thesis suggests that advancing manipulation under complex dynamics may not require perfect physical models or complete observations. Instead, systems might succeed by modeling just enough: structuring action in physically meaningful ways, and extracting perceptual signals that support generalization and adaptation. By continuing to explore how these elements can complement each other, future work may bring us closer to manipulation systems that are both more capable and more adaptable under complex dynamics.

# Bibliography


[1] Tali Basha, Yael Moses, and Nahum Kiryati. Multi-view scene flow estimation: A view centered variational approach. *International Journal of Computer Vision*, 101:6–21, 2010. URL https://api.semanticscholar.org/CorpusID:1284146. 3.2.3

[2] Steven S. Beauchemin and John L. Barron. The computation of optical flow. *ACM computing surveys (CSUR)*, 27(3):433–466, 1995. 3.2.3

[3] Homanga Bharadhwaj, Roozbeh Mottaghi, Abhinav Gupta, and Shubham Tulsiani. Track2act: Predicting point tracks from internet videos enables diverse zero-shot robot manipulation, 2024. 3.1, 3.2.4

[4] Aditya Bhatt, Adrian Sieler, Steffen Puhlmann, and Oliver Brock. Surprisingly robust in-hand manipulation: An empirical study. *arXiv preprint arXiv:2201.11503*, 2022. 2.1, 2.2.3

[5] Diego Bianchi, Michele Gabrio Antonelli, Cecilia Laschi, Angelo Maria Sabatini, and Egidio Falotico. Softoss: Learning to throw objects with a soft robot. *IEEE Robotics & Automation Magazine*, 2023. 2.2.1

[6] Diego Bianchi, Giulia Campinoti, Costanza Comitini, Cecilia Laschi, Alessandro Rizzo, Angelo Maria Sabatini, and Egidio Falotico. Softsling: A soft robotic arm control strategy to throw objects with circular run-ups. *IEEE Robotics and Automation Letters*, 2024. 2.2.3

[7] Michael Bombile and Aude Billard. Dual-arm control for coordinated fast grabbing and tossing of an object: Proposing a new approach. *IEEE Robotics & Automation Magazine*, 29(3):127–138, 2022. 2.1, 2.2.1

[8] Aljaž Božič, Michael Zollhöfer, Christian Theobalt, and Matthias Nießner. Deepdeform: Learning non-rigid rgb-d reconstruction with semi-supervised data, 2020. 3.2.3

[9] Aljaž Božič, Pablo Palafox, Michael Zollhöfer, Angela Dai, Justus Thies, and Matthias Nießner. Neural non-rigid tracking, 2021. 3.2.3

[10] Daniel Bruder, C David Remy, and Ram Vasudevan. Nonlinear system identifica-

tion of soft robot dynamics using koopman operator theory. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6244–6250. IEEE, 2019. 2.2.3

[11] 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. 3.2.2, 3.4.2, 3.5

[12] Chen Cao, Yanlin Weng, Stephen Lin, and Kun Zhou. 3d shape regression for real-time facial animation. *ACM Transactions on Graphics (TOG)*, 32:1 – 10, 2013. URL https://api.semanticscholar.org/CorpusID:2818777. 3.2.3

[13] Lawrence Yunliang Chen, Huang Huang, Ellen Novoseller, Daniel Seita, Jeffrey Ichnowski, Michael Laskey, Richard Cheng, Thomas Kollar, and Ken Goldberg. Efficiently learning single-arm fling motions to smooth garments. In *The International Symposium of Robotics Research*, pages 36–51. Springer, 2022. 2.2.1

[14] Tao Chen, Jie Xu, and Pulkit Agrawal. A system for general in-hand object re-orientation. In *Conference on Robot Learning*, pages 297–307. PMLR, 2022. 2.1

[15] Ho Kei Cheng and Alexander G. Schwing. Xmem: Long-term video object segmentation with an atkinson-shiffrin memory model, 2022. 3.6.1

[16] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023. 1

[17] Cheng Chi, Benjamin Burchfiel, Eric Cousineau, Siyuan Feng, and Shuran Song. Iterative residual policy: for goal-conditioned dynamic manipulation of deformable objects. *The International Journal of Robotics Research*, 43(4): 389–404, 2024. 2.2.1

[18] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. URL http://www.blender.org. (document), 3.4

[19] Xinke Deng, Yu Xiang, Arsalan Mousavian, Clemens Eppner, Timothy Bretl, and Dieter Fox. Self-supervised 6d object pose estimation for robot manipulation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3665–3671. IEEE, 2020. 3.1, 3.2.4

[20] Shivin Devgon, Jeffrey Ichnowski, Ashwin Balakrishna, Harry Zhang, and Ken Goldberg. Orienting novel 3d objects using self-supervised learning of rotation transforms. In *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, pages 1453–1460. IEEE, 2020. 3.1, 3.2.4

[21] Carl Doersch, Ankush Gupta, Larisa Markeeva, Adrià Recasens, Lucas Smaira,

Yusuf Aytar, João Carreira, Andrew Zisserman, and Yi Yang. Tap-vid: A benchmark for tracking any point in a video. *Advances in Neural Information Processing Systems*, 35:13610–13626, 2022. 3.2.3

[22] Carl Doersch, Ankush Gupta, Larisa Markeeva, Adrià Recasens, Lucas Smaira, Yusuf Aytar, João Carreira, Andrew Zisserman, and Yi Yang. Tap-vid: A benchmark for tracking any point in a video, 2023. 3.1

[23] 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. *arXiv preprint arXiv:2306.08637*, 2023. 3.2.3

[24] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B. Tenenbaum, and Jiajun Wu. Neural radiance flow for 4d view synthesis and video processing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 3.2.2

[25] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. In *SIGGRAPH Asia 2022 Conference Papers*, 2022. 3.4.2, 3.5

[26] 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. 3.2.2, 3.5

[27] Zipeng Fu, Tony Z Zhao, and Chelsea Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024. 1

[28] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *Proceedings of the IEEE International Conference on Computer Vision*, 2021. 3.2.2

[29] 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 (ICCV)*, pages 16022–16033, October 2023. 3.2.3

[30] Huy Ha and Shuran Song. Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding. In *Conference on Robot Learning*, pages 24–33. PMLR, 2022. 2.1, 2.2.1

[31] David A Haggerty, Michael J Banks, Ervin Kamenar, Alan B Cao, Patrick C Curtis, Igor Mezić, and Elliot W Hawkes. Control of soft robots with inertial dynamics. *Science robotics*, 8(81):eadd6864, 2023. 2.2.3

[32] N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 312–317, 1996. doi: 10.1109/ICEC.1996.542381. 2.4.5, 2.5.1

[33] Adam W Harley, Zhaoyuan Fang, and Katerina Fragkiadaki. Particle video revisited: Tracking through occlusions using point trajectories. In *European Conference on Computer Vision*, pages 59–75. Springer, 2022. 3.2.3

[34] Bianca S Homberg, Robert K Katzschmann, Mehmet R Dogar, and Daniela Rus. Robust proprioceptive grasping with a soft robot hand. *Autonomous robots*, 43: 681–696, 2019. 2.1

[35] Binghao Huang, Yuanpei Chen, Tianyu Wang, Yuzhe Qin, Yaodong Yang, Nikolay Atanasov, and Xiaolong Wang. Dynamic handover: Throw and catch with bimanual hands. *arXiv preprint arXiv:2309.05655*, 2023. 1

[36] Jeffrey Ichnowski, Michael Danielczuk, Jingyi Xu, Vishal Satish, and Ken Goldberg. GOMP: Grasp-optimized motion planning for bin picking. In *2020 IEEE international conference on robotics and automation (ICRA)*, pages 5270–5277. IEEE, 2020. 2.1

[37] Jeffrey Ichnowski, Yahav Avigal, Yi Liu, and Ken Goldberg. GOMP-FIT: Grasp-optimized motion planning for fast inertial transport. In *2022 international conference on robotics and automation (ICRA)*, pages 5255–5261. IEEE, 2022. 2.1, 2.2.1

[38] Tatsuya Ishihara, Akio Namiki, Masatoshi Ishikawa, and Makoto Shimojo. Dynamic pen spinning using a high-speed multifingered hand with high-speed tactile sensor. In *2006 6th IEEE-RAS International Conference on Humanoid Robots*, pages 258–263. IEEE, 2006. 2.2.2

[39] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker: It is better to track together, 2023. 3.1

[40] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George 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/`. 3.1, 3.2.1, 3.4.1, 3.5

[41] Leonid Keselman and Martial Hebert. Approximate differentiable rendering with algebraic surfaces. In *European Conference on Computer Vision (ECCV)*, 2022. 3.1, 3.2.1

[42] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything, 2023. 3.6.1

[43] Z. Li, S. Niklaus, N. Snavely, and O. Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6494–6504, Los Alamitos, CA, USA, jun 2021. IEEE Computer Society. doi: 10.1109/CVPR46437.2021.00643. URL https://doi.ieeecomputersociety.org/10.1109/CVPR46437.2021.00643. 3.2.2

[44] Litian Liang, Liuyu Bian, Caiwei Xiao, Jialin Zhang, Linghao Chen, Isabella Liu, Fanbo Xiang, Zhiao Huang, and Hao Su. Robo360: A 3d omnispective multi-material robotic manipulation dataset, 2023. (document), 3.1, 3.6.3, 3.6

[45] Vincent Lim, Huang Huang, Lawrence Yunliang Chen, Jonathan Wang, Jeffrey Ichnowski, Daniel Seita, Michael Laskey, and Ken Goldberg. Planar robot casting with real2sim2real self-supervised learning. *arXiv preprint arXiv:2111.04814*, 2021. 2.1

[46] Jiehong Lin, Lihua Liu, Dekun Lu, and Kui Jia. Sam-6d: Segment anything model meets zero-shot 6d object pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 27906–27916, 2024. 3.1, 3.2.4

[47] Jia-Wei Liu, Yan-Pei Cao, Weijia Mao, Wenqiao Zhang, David Junhao Zhang, Jussi Keppo, Ying Shan, Xiaohu Qie, and Mike Zheng Shou. Devrf: Fast deformable voxel radiance fields for dynamic scenes. *Advances in Neural Information Processing Systems*, 35:36762–36775, 2022. 3.2.2, 3.2.3

[48] Yang Liu and Aude Billard. Tube acceleration: robust dexterous throwing against release uncertainty. *IEEE Transactions on Robotics*, 2024. 2.2.1

[49] Yang Liu, Uksang Yoo, Seungbeom Ha, S Farokh Atashzar, and Farshid Alambeigi. Influence of antagonistic tensions on distributed friction forces of multisegment tendon-driven continuum manipulators with irregular geometry. *IEEE/ASME Transactions on Mechatronics*, 27(5):2418–2428, 2021. 2.1

[50] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *arXiv preprint arXiv:2308.09713*, 2023. 3.1, 3.2.2, 3.2.3, 3.5.1, 3.6.1, 3.6.2

[51] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023. 1, 2.2.2

[52] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 3.2.1

[53] Andrew S Morgan, Bowen Wen, Junchi Liang, Abdeslam Boularias, Aaron M

Dollar, and Kostas Bekris. Vision-driven compliant manipulation for reliable, high-precision assembly tasks. *RSS*, 2021. 3.2.4

[54] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. doi: 10.1145/3528223.3530127. URL https://doi.org/10.1145/3528223.3530127. 3.5

[55] Yuzuko C Nakamura, Daniel M Troniak, Alberto Rodriguez, Matthew T Mason, and Nancy S Pollard. The complexities of grasping in the wild. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 233–240. IEEE, 2017. 2.1

[56] Shoma Nakatani and Yuji Yamakawa. Dynamic manipulation like normal-type pen spinning by a high-speed robot hand and a high-speed vision system. In *2023 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 636–642. IEEE, 2023. 2.1, 2.2.2

[57] Richard A. Newcombe, Dieter Fox, and Steven M. Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 343–352, 2015. doi: 10.1109/CVPR.2015.7298631. 3.2.3

[58] Iasonas Oikonomidis, Nikolaos Kyriazis, and Antonis A. Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *British Machine Vision Conference*, 2011. URL https://api.semanticscholar.org/CorpusID:8677556. 3.2.3

[59] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024. 3.1

[60] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. *ICCV*, 2021. 3.2.2

[61] Steffen Puhlmann, Jason Harris, and Oliver Brock. RBO hand 3: A platform for soft dexterous manipulation. *IEEE Transactions on Robotics*, 38(6):3434–3449, 2022. 2.2.3

[62] 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. 3.4.2

[63] 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*, 2021. 3.2.2

[64] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. 2.4.4

[65] Tanner Schmidt, Richard A. Newcombe, and Dieter Fox. Dart: dense articulated real-time tracking with consumer depth cameras. *Autonomous Robots*, 39:239 – 258, 2015. URL https://api.semanticscholar.org/CorpusID:254251877. 3.2.3

[66] Kenneth Shaw, Yulong Li, Jiahui Yang, Mohan Kumar Srirama, Ray Liu, Haoyu Xiong, Russell Mendonca, and Deepak Pathak. Bimanual dexterity for complex tasks. In *8th Annual Conference on Robot Learning*, 2024. 1

[67] Haochen Shi, Huazhe Xu, Zhiao Huang, Yunzhu Li, and Jiajun Wu. Robocraft: Learning to see, simulate, and shape elasto-plastic objects with graph networks, 2022. 3.1

[68] Adrian Sieler and Oliver Brock. Dexterous soft hands linearize feedback-control for in-hand manipulation. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8757–8764. IEEE, 2023. 2.1

[69] Nina R Sinatra, Clark B Teeple, Daniel M Vogt, Kevin Kit Parker, David F Gruber, and Robert J Wood. Ultragentle manipulation of delicate structures using a soft robotic gripper. *Science Robotics*, 4(33):eaax5425, 2019. 2.1

[70] Liangchen Song, Xuan Gong, Benjamin Planche, Meng Zheng, David Doermann, Junsong Yuan, Terrence Chen, and Ziyan Wu. Pref: Predictability regularized neural motion fields. In *European Conference on Computer Vision*, 2022. 3.2.3

[71] Balakumar Sundaralingam and Tucker Hermans. Geometric in-hand regrasp planning: Alternating optimization of finger gaits and in-grasp manipulation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 231–238. IEEE, 2018. 2.1

[72] Marwan Taher, Ignacio Alzugaray, and Andrew J Davison. Fit-ngp: Fitting object models to neural graphics primitives. *arXiv preprint arXiv:2401.02357*, 2024. 3.1, 3.2.4

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

3.2.3, 3.6.1

[74] Mel Vecerik, Carl Doersch, Yi Yang, Todor Davchev, Yusuf Aytar, Guangyao Zhou, Raia Hadsell, Lourdes Agapito, and Jon Scholz. Robotap: Tracking arbitrary points for few-shot visual imitation. *arXiv preprint arXiv:2308.15975*, 2023. 3.2.4

[75] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 722–729. IEEE, 1999. 3.2.3

[76] Christoph Vogel, Konrad Schindler, and Stefan Roth. 3d scene flow estimation with a piecewise rigid scene model. *International Journal of Computer Vision*, 115:1–28, 2015. URL https://api.semanticscholar.org/CorpusID:3358790. 3.2.3

[77] Chaoyang Wang, Ben Eckart, Simon Lucey, and Orazio Gallo. Neural trajectory fields for dynamic novel view synthesis, 2021. 3.2.3

[78] Jun Wang, Ying Yuan, Haichuan Che, Haozhi Qi, Yi Ma, Jitendra Malik, and Xiaolong Wang. Lessons from learning to spin "pens". In *8th Annual Conference on Robot Learning*. 2.1, 2.2.2

[79] Qianqian Wang, Yen-Yu Chang, Ruojin Cai, Zhengqi Li, Bharath Hariharan, Aleksander Holynski, and Noah Snavely. Tracking everything everywhere all at once. In *International Conference on Computer Vision*, 2023. 3.2.3, 3.6.1

[80] Bowen Wen, Chaitanya Mitash, Baozhang Ren, and Kostas E Bekris. se (3)-tracknet: Data-driven 6d pose tracking by calibrating image residuals in synthetic domains. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10367–10373. IEEE, 2020. 3.1, 3.2.4

[81] Bowen Wen, Wenzhao Lian, Kostas Bekris, and Stefan Schaal. You only demonstrate once: Category-level manipulation from single visual demonstration. *RSS*, 2022. 3.2.4

[82] Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. Foundationpose: Unified 6d pose estimation and tracking of novel objects, 2024. 3.1, 3.2.4

[83] Chuan Wen, Xingyu Lin, John So, Kai Chen, Qi Dou, Yang Gao, and Pieter Abbeel. Any-point trajectory modeling for policy learning, 2023. 3.1, 3.1, 3.2.4

[84] 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. *arXiv preprint arXiv:2310.08528*, 2023. (document), 3.1, 3.2.2, 3.4.2, 3.5, 3.6.1, 3.1, 3.6.4, 3.7

[85] W. Xian, J. Huang, J. Kopf, and C. Kim. Space-time neural irradiance fields for free-viewpoint video. In *2021 IEEE/CVF Conference on Computer Vision and*

*Pattern Recognition (CVPR)*, pages 9416–9426, Los Alamitos, CA, USA, jun 2021. IEEE Computer Society. doi: 10.1109/CVPR46437.2021.00930. URL https://doi.ieeecomputersociety.org/10.1109/CVPR46437.2021.00930. 3.2.2

[86] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017. 3.1, 3.2.4

[87] Yuji Yamakawa, Akio Namiki, and Masatoshi Ishikawa. Dynamic high-speed knotting of a rope by a manipulator. *International Journal of Advanced Robotic Systems*, 10(10):361, 2013. 2.2.1

[88] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics*, 38(6):1–14, November 2019. ISSN 1557-7368. doi: 10.1145/3355089.3356513. URL http://dx.doi.org/10.1145/3355089.3356513. 3.4.1

[89] Uksang Yoo, Nathaniel Dennler, Maja Mataric, Stefanos Nikolaidis, Jean Oh, and Jeffrey Ichnowski. MOE-hair: Toward soft and compliant contact-rich hair manipulation and care. In *Companion of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, pages 1163–1167, 2024. 2.1, 2.1

[90] Uksang Yoo, Ziven Lopez, Jeffrey Ichnowski, and Jean Oh. POE: Acoustic soft robotic proprioception for omnidirectional end-effectors. *arXiv preprint arXiv:2401.09382*, 2024. 2.4.1

[91] Uksang Yoo, Nathaniel Dennler, Eliot Xing, Maja Matarić, Stefanos Nikolaidis, Jeffrey Ichnowski, and Jean Oh. Soft and compliant contact-rich hair manipulation and care. In *Proceedings of the 2025 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '25, page 610–619. IEEE Press, 2025. 2.4.1

[92] Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*, 36(4):1307–1319, 2020. 2.2.1

[93] Harry Zhang, Jeffrey Ichnowski, Daniel Seita, Jonathan Wang, Huang Huang, and Ken Goldberg. Robots of the lost arc: Self-supervised learning to dynamically manipulate fixed-endpoint cables. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4560–4567. IEEE, 2021. 2.1, 2.2.1

[94] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023. 1

[95] Yang Zheng, Adam W Harley, Bokui Shen, Gordon Wetzstein, and Leonidas J Guibas. Pointodyssey: A large-scale synthetic dataset for long-term point tracking. In *Proceedings of the IEEE/CVF International Conference on Computer*

*Vision*, pages 19855–19865, 2023. 3.2.3, 3.6.2

[96] Michael Zollhöfer, Matthias Nießner, Shahram Izadi, Christoph Rhemann, Christopher Zach, Matthew Fisher, Chenglei Wu, Andrew William Fitzgibbon, Charles T. Loop, Christian Theobalt, and Marc Stamminger. Real-time non-rigid reconstruction using an rgb-d camera. *ACM Transactions on Graphics (TOG)*, 33:1 – 12, 2014. URL https://api.semanticscholar.org/CorpusID:9616070. 3.2.3