SplatSim: Zero-Shot Sim2Real Transfer of RGB Manipulation Policies Using Gaussian Splatting

Mohammad Nomaan Qureshi

CMU-RI-TR-25-54

June 05, 2025

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

Thesis Committee:
Prof. George Kantor, *co-chair*
Prof. Abhisesh Silwal, *co-chair*
Prof. Shubham Tulsiani
John Kim,

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

To my Parents.

Abstract

Sim2Real transfer, particularly for manipulation policies relying on RGB images, remains a critical challenge in robotics due to the significant domain shift between synthetic and real-world visual data. In this work, we propose *SplatSim*, a novel framework that leverages Gaussian Splatting as the primary rendering primitive to reduce the Sim2Real gap for RGB-based manipulation policies. By replacing traditional mesh representations with Gaussian Splats in simulators, *SplatSim* produces highly photorealistic synthetic data while maintaining the scalability and cost-efficiency of simulation. We demonstrate the effectiveness of our framework by training manipulation policies within *SplatSim* and deploying them in the real world in a zero-shot manner, achieving an average success rate of 86.25%, compared to 97.5% for policies trained on real-world data.

I would like to express my sincere gratitude to everyone who supported me throughout my journey at Carnegie Mellon University. Specifically, I want to thank Prof. George Kantor and Dr. Abhisesh Silwal for their constant support and invaluable guidance. Their mentorship played a crucial role in shaping my academic and professional development, and I'm truly grateful for his contributions. Additionally, I would also like to extend my thanks to Dr. Francisco Yandun and Prof. David Held for their excellent mentorship, which helped me broaden my knowledge and develop new skills. I would also like to express my appreciation towards Sparsh Garg, Vedant Mundheda and John Kim for their support and encouragement throughout my time at CMU. Their contributions have been instrumental in making my journey at CMU a truly enriching experience.

I would like to express my heartfelt gratitude to my parents, who have made countless sacrifices over the years to provide me with a quality education. Their unwavering support and encouragement have been a constant source of strength and inspiration for me, and I cannot thank them enough for everything they have done. I am also immensely grateful to my beloved sisters, who have always been there for me whenever I needed them. Their love and kindness have meant the world to me, and I feel blessed to have them in my life. Lastly, I want to extend my thanks to all the other members of my extended family for their love and support.

Contents

1 Introduction **1**

2 Related Works **5**
2.1 Sim2real 5
2.2 Gaussian Splatting for Robotics 6

3 Method **7**
3.1 Problem Statement 8
3.2 Definitions of Coordinate Frames and Transformations 9
3.3 Robot Splat Models 9
3.4 Object Splat Models 11
3.5 Articulated Object 11
3.6 Rendering Simulated Trajectories using SplatSim 12
3.7 Policy Training and Deployment 12

4 Experiments **15**
4.1 Demonstrations in the Real World and Simulation 15
4.2 Zero-Shot Policy Deployment Results 16
4.3 Quantifying Robot Renderings 18
4.4 Effect of Augmentations 19

5 Conclusions **21**
5.1 Conclusions 21

A Extending SplatSim to Articulated and Deformable objects **23**
A.1 Pipeline for Articulated Objects 23
A.1.1 Robot Link Segmentation Using KNN Classifier 23
A.1.2 Forward Kinematics Transformation for Articulated Links . . 24
A.1.3 Rendering Articulated Objects 25
A.2 Segmentation of Cloth Gaussians into Mesh Faces 25
A.2.1 2D Projection of Mesh Vertices and Points 25
A.2.2 Point-in-Triangle Test Using Barycentric Coordinates 25
A.3 Deformation of Cloth with Rigid Body Approximation 26
A.3.1 Retrieving Vertex Data 26

A.3.2 Centroid Calculation 26
A.3.3 Aligning Faces Using SVD 27
A.3.4 Calculating Translation 27
A.3.5 Constructing Transformation Matrix 27
A.4 Limitations of the Current Approach 27
A.5 Experiments 28

B Applying Scene Modelling to the Field Setting **29**
B.1 Motivation 29
B.2 System Overview 30
B.3 Robot Localization via Visual Servoing in Gaussian Splat Space . . . 31
B.4 Pregrasp Planning and Execution 32
B.4.1 Discussion and Conclusion 33

Bibliography **35**

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

1.1 We employ Gaussian Splatting [23] as the primary rendering primitive within existing simulation environments to generate highly photorealistic synthetic data for robotic manipulation tasks. Our framework retains all the traditional advantages of simulators—including scalability, cost-efficiency, and safety—while enhancing visual realism. Policies trained exclusively on this synthetic data exhibit zero-shot transfer capabilities to real-world scenarios, achieving performance comparable to those trained on real-world datasets. 2

3.1 **Top:** Our proposed SplatSim framework. Expert demonstrations are collected (a) in a physics simulator (PyBullet). In our case, these demonstrations come either from human experts (teleoperation via Gello [46]) or through a privileged information-based motion planner. The trajectories from the simulator are then fed to the simulator-aligned splat models of the scene and the object (b). We transform the 3D Gaussians to manipulate the static Gaussian Splat models, as delineated in Sec. 3.3, to extract photorealistic renderings of the scene at novel joint and object poses, which serve as the RGB state observations for the diffusion policy. Along with these RGB observations, diffusion policy [9] also takes the end effector position and orientation as the input. We augment the end effector states as well. **Bottom:** Once trained with the sim data, we freeze the policy and directly deploy it to the real-world setting. 8

3.2 The robot is visualized in a static scene by first creating a Gaussian splat of the scene with the robot in its home position. The robot's point cloud is manually segmented and aligned with the canonical robot frame using the ICP algorithm. Each robot link is then segmented, and forward kinematics transformations are applied, enabling the rendering of the robot at arbitrary joint configurations. 10

3.3 We use a KNN-based classifier for segmenting links for articulated objects like parallel jaw grippers. We train a KNN model with the ground truth point labeling from the URDF model of the end effector. 12

4.1 SplatSim Rollout: Renderings from our SplatSim framework across four different manipulation tasks. 17

A.1 The figure illustrates temporal sequences for different object manipulations. (A) Demonstrates the progression of articulated object manipulation, showcasing the movement of eyeglass handles over time. (B) Presents the time steps involved in soft object manipulation, specifically depicting the folding process of a cloth. (C) Displays the limitation of the current method of rendering clothes for extreme deformations. . 24

B.1 Overview of the field deployment pipeline. A raw Gaussian Splat is first scaled using a known calibration object. The robot is then localized in the splat frame via servoing or registration. Once localized, expert-annotated pruning points in the splat can be transformed into the robot frame, enabling planning and execution of pruning actions using simple Cartesian motion. The system supports both open-loop and closed-loop execution. 31

List of Tables

4.1 Comparison of task success rates and data collection times across various manipulation tasks. Our policies trained solely on synthetic data achieve an 86.25% zero-shot Sim2Real performance, comparable to those trained on real-world data. By leveraging the automation capabilities of simulators, we significantly reduce the human effort required for data generation. 16

Chapter 1

Introduction

The Sim2Real problem, a focal challenge in robotics, pertains to the transfer of control policies learned in simulated environments to real world settings. Recently, significant progress has been made in deploying controllers trained in simulation to the real world in a zero-shot manner. Robots have demonstrated the ability to walk on rough terrains [2, 8], perform in-hand object rotation [16, 35, 48, 51], and grasp previously unseen objects [29]. Notably, all of these methods rely on perception modalities like depth, tactile sensing, or point cloud inputs, which have gained significant attention due to the relatively small Sim2Real gap they offer. The reduced discrepancy between simulated and real-world data in these modalities has led to remarkable progress, reinforcing the idea that *policies trained on modalities that can be simulated well, can be transferred well.*

In contrast, RGB images are rarely used as the primary sensing modality in robot learning applications. RGB images offer several unique advantages over other commonly used modalities in Sim2Real transfer. They capture crucial visual details such as color, texture, lighting, and surface reflectivity, to name a few, which are essential for understanding complex environments. For instance, in a simple task of plucking ripe fruits, color is a key feature for determining ripeness—an inference straightforward in RGB space but difficult and impractical with depth or tactile inputs. Additionally, RGB images are easy to acquire in real-world environments with cameras and align closely with human perception, making them well-suited for interpreting intricate details in dynamic and complex scenes.

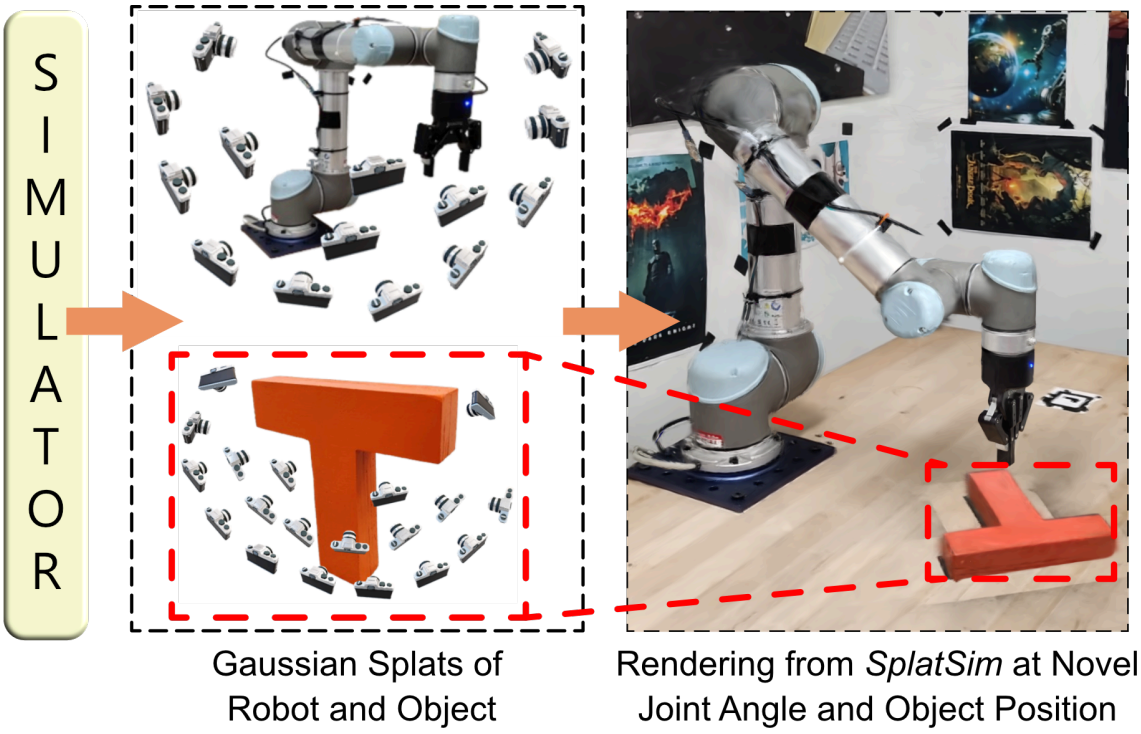

Figure 1.1: We employ Gaussian Splatting [23] as the primary rendering primitive within existing simulation environments to generate highly photorealistic synthetic data for robotic manipulation tasks. Our framework retains all the traditional advantages of simulators—including scalability, cost-efficiency, and safety—while enhancing visual realism. Policies trained exclusively on this synthetic data exhibit zero-shot transfer capabilities to real-world scenarios, achieving performance comparable to those trained on real-world datasets.

But why has it been difficult to deploy policies trained in simulation with RGB information to the real world? The problem lies in the fact that the distribution of images the robot observes in the simulator is very different from the distribution of images it would see in the real world. This makes "vision Sim2Real an out-of-domain generalization problem" [50], a fundamental challenge in machine learning that is still unsolved. For this reason, policies trained on simulated images often struggle when applied to distributions of real-world images.

In this work, we propose a systematic and novel method to reduce the Sim2Real gap for RGB images, by leveraging Gaussian Splatting [23] as a photorealistic render, using existing simulators as the physics backbone. We propose utilizing Gaussian Splatting [23] as the primary rendering primitive, replacing traditional mesh-based representations in existing simulators, to significantly improve the photo-realism of

rendered scenes. By integrating these renderings of simulated demonstrations with state-of-the-art behavior cloning techniques, we introduce a framework for zero-shot transfer of manipulation policies trained entirely on simulation data, to the real world. Our key contributions are as follows:

- We propose a novel and scalable data generation framework, "*SplatSim*" for manipulation tasks. *SplatSim* is focused predominantly on bridging the vision Sim2Real gap by leveraging photorealistic renderings generated through Gaussian Splatting, replacing traditional mesh representation in the rendering pipeline of the simulator.
- We leverage Robot Splat Models and Object Splat Models with a simulator as a physics backend to generate photorealistic trajectories of robot-object interactions. Our method eliminates the need for real-world data collection, relying only on an initial video of the static scene. We demonstrate how these renderings, combined with simulated demonstrations, enable high-quality synthetic datasets for behavior cloning.
- We demonstrate the effectiveness of our framework by deploying RGB policies, trained entirely in simulation, to the real world in a zero-shot manner across four tasks, achieving an average success rate of 86.25%, compared to 97.5% for policies trained on the real-world data.

Chapter 2

Related Works

2.1 Sim2real

Robotics simulation tools like [3, 11, 15, 32, 33, 41, 43] have become invaluable in scaling up robot learning due to several advantages including parallelization, cost and time efficiency, and safety. Recent advancements in transferring learned policies from simulation to the real world have demonstrated impressive results, particularly in domains that leverage modalities with a low Sim2Real gap, such as depth, point cloud, proprioception, or tactile feedback. These modalities have enabled robots to perform contact-rich tasks like quadruped locomotion [2, 8, 14, 18, 55] and bipedal locomotion [25, 37], dexterous manipulation [16, 29, 35, 48, 51], manipulation of articulated objects [13, 53], among others [19, 26, 40, 54].

However, Sim2Real transfer for RGB-based manipulation policies remains challenging. Several prior works [20, 21, 30, 42] have explored Sim2Real transfer for RGB-based manipulation policies using domain randomization techniques. These approaches, however, often require task-specific tuning and environment engineering, which can be both labor-intensive and difficult to achieve accurately in traditional physics simulators. In contrast, our method eliminates the need for task-specific adjustments and leverages an initial scan of the target deployment environment, significantly simplifying the process. Other existing approaches such as domain adaptation [38], often rely on extensive offline data collection of real-world object interactions. Our method requires only an initial video of the static scene without

the need for additional real-world data collection.

In this work, we address the challenge of transferring RGB-based policies for manipulation tasks using Gaussian Splatting, which requires rendering complex interactions between objects and the robot. A work notably related to ours is RialTo [44] which uses a Real2Sim2Real approach similar to ours. However, their policy is still trained on point clouds, which requires depth during execution time. In contrast, *SplatSim* only uses RGB images for learning and policy deployment. Another recent work Maniwhere [52] does large-scale reinforcement learning in simulation and shows generalization to the real world, however, their method still requires depth at test time and cannot work with just RGB images in the real world.

2.2 Gaussian Splatting for Robotics

Gaussian Splatting [23] is a state-of-the-art rendering technique that models scenes using 3D Gaussian primitives, offering an efficient and photorealistic representation of complex geometries. In contrast to NeRF [31] and its derivatives [5, 34, 49], the explicit, point cloud-like structure of Gaussian Splats enables easier manipulation, which has led to numerous subsequent works focused on dynamic Gaussian Splatting models [4, 7, 28, 45, 47]. This explicit nature of Gaussian Splatting has also garnered interest in the robotics community, with recent studies applying it to language-guided manipulation [39], object grasping [22], and deformable object manipulation [12]. The two related works, Embodied Gaussians [1] and RoboStudio [27], focus on learning from real-world data. Embodied Gaussians [1] directly learns a forward model for robot-object interactions, requiring real-world data for each new robot and object, while we offload dynamics to a physics engine and focus on RGB-based policy deployment. RoboStudio [27] combines simulation with Gaussian Splatting but focuses on system identification, whereas our approach generates synthetic data for real-world deployment using existing simulators. Another closely related work to our method is [36], which combines Gaussian Splatting with a simulator, but it is focused on navigation. Unlike manipulation tasks, the agent in their approach does not interact with or manipulate the environment.

Chapter 3

Method

The key premise of our method is that if each rigid body in the Gaussian Splat representation of the real-world scene can be accurately segmented, and its corresponding homogeneous transformation relative to the simulator is identified, then it becomes feasible to render the rigid body in novel poses. The rigid bodies can include links of the robot, links of the gripper, articulated objects, or simple non-deformable objects. By applying this process to all rigid bodies interacting with the robot in simulation, we can generate photorealistic renderings for an entire demonstration trajectory. This approach is analogous to traditional rendering in simulators; however, instead of using mesh primitives, we utilize Gaussian Splats as the underlying representation. This approach allows us to be more effective at capturing the detailed visual fidelity of real-world scenes.

The following subsections describe our method. We begin by formalizing the problem statement in Sec. 3.1 and notations in Sec. 3.2. Next, we detail the segmentation and rendering of each robot link at novel joint poses in Sec. 3.3, individual objects at new positions in Sec. 3.4, and grippers in Sec. 3.5. Sec. 3.6 covers the rendering of complete robot-object interaction trajectories, followed by the policy training protocol in Sec. 3.7.

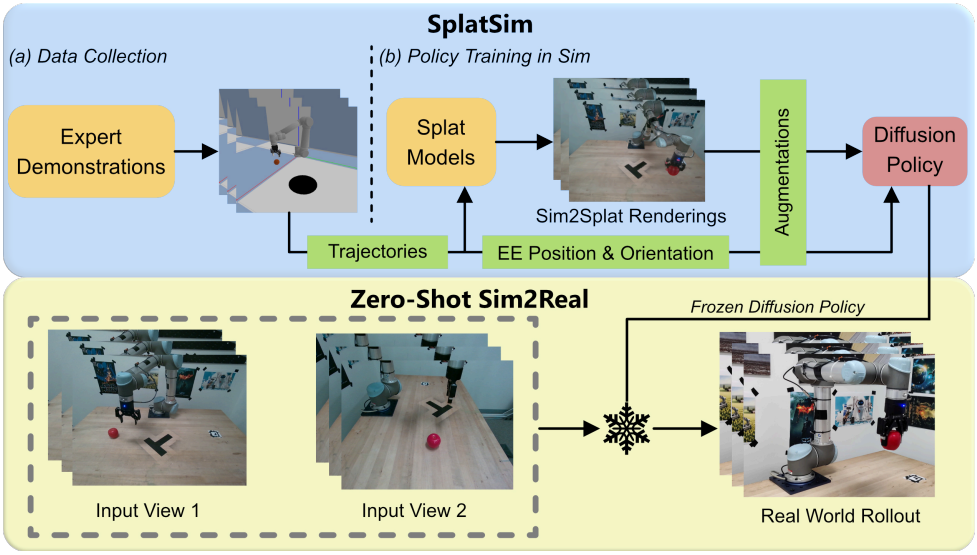

Figure 3.1: **Top:** Our proposed SplatSim framework. Expert demonstrations are collected (a) in a physics simulator (PyBullet). In our case, these demonstrations come either from human experts (teleoperation via Gello [46]) or through a privileged information-based motion planner. The trajectories from the simulator are then fed to the simulator-aligned splat models of the scene and the object (b). We transform the 3D Gaussians to manipulate the static Gaussian Splat models, as delineated in Sec. 3.3, to extract photorealistic renderings of the scene at novel joint and object poses, which serve as the RGB state observations for the diffusion policy. Along with these RGB observations, diffusion policy [9] also takes the end effector position and orientation as the input. We augment the end effector states as well. **Bottom:** Once trained with the sim data, we freeze the policy and directly deploy it to the real-world setting.

3.1 Problem Statement

We define $\mathcal{S}_{real}$ as the Gaussian Splat of a real-world scene, captured from multiple RGB viewpoints, including the robot. We also define $\mathcal{S}^k_{obj}$ as the splat of the k-th object in the scene, captured from multiple viewpoints. Our goal is to use $\mathcal{S}_{real}$ for generating photorealistic renderings I^{sim} of a robot operating in any simulator (e.g., PyBullet). Then, we can leverage this representation to collect demonstrations using the expert $\mathcal{E}$ for training RGB-based policies.

The expert $\mathcal{E}$ generates a trajectory $\tau_{\mathcal{E}}$ consisting of state-action pairs $\{(s_1, a_1), \ldots, (s_T, a_T)\}$ for a full episode. The state at each time step t is defined as $s_t = (q_t, x^1_t, \ldots, x^n_t)$,

where $q_t \in \mathbb{R}^m$ denotes the robot's joint angles and $x_t^k = (p_t^k, R_t^k)$ represents the position $p_t^k \in \mathbb{R}^3$ and orientation $R_t^k \in SO(3)$ of the k-th object in the scene. The corresponding action $a_t = (p_t^e, R_t^e)$ refers to the end effector's position $p_t^e \in \mathbb{R}^3$ and orientation $R_t^e \in SO(3)$.

The renderings I^{sim}, derived from these simulated states s_t, are used as inputs to train the policy $\pi_{\mathcal{I}}$. The policy relies solely on real-world RGB images I^{real} at test time.

3.2 Definitions of Coordinate Frames and Transformations

We define several coordinate frames to clarify the relationships between the real-world scene, the simulator, and the splat point clouds. The real-world coordinate frame, denoted as $\mathcal{F}_{real}$, serves as the primary reference frame. Both the simulator coordinate frame, $\mathcal{F}_{sim}$, and the real-world robot frame, $\mathcal{F}_{robot}$, are aligned with $\mathcal{F}_{real}$. This alignment ensures that the robot's base in the simulator and the real world share the same coordinate system.

Additionally, the splat coordinate frame, denoted as $\mathcal{F}_{splat}$, represents the frame of the base of the robot in the Gaussian Splat of the scene $\mathcal{S}_{real}$. The robot base in the splat point cloud has a different frame from $\mathcal{F}_{real}$, and we account for this difference by using the transformation matrix $T_{\mathcal{F}_{robot}}^{\mathcal{F}_{splat}}$.

We also define the k-th object frame in the simulator, $\mathcal{F}_{k-obj,sim}$, where objects are initialized in $\mathcal{SIM}$ at the origin with no rotation. The k-th object frame in the splat, $\mathcal{F}_{k-obj,splat}$, represents the object's position and orientation in its Gaussian splat $\mathcal{S}_{obj}$. The object frames $\mathcal{F}_{k-obj,sim}$ and $\mathcal{F}_{k-obj,splat}$ are later aligned during the simulation and splat process using the transformation matrix $T_{\mathcal{F}_{k-obj,splat}}^{\mathcal{F}_{k-obj,sim}}$.

3.3 Robot Splat Models

Our method for obtaining robot renderings at novel joint poses is summarized in Fig. 3.2. It follows a three-step approach:

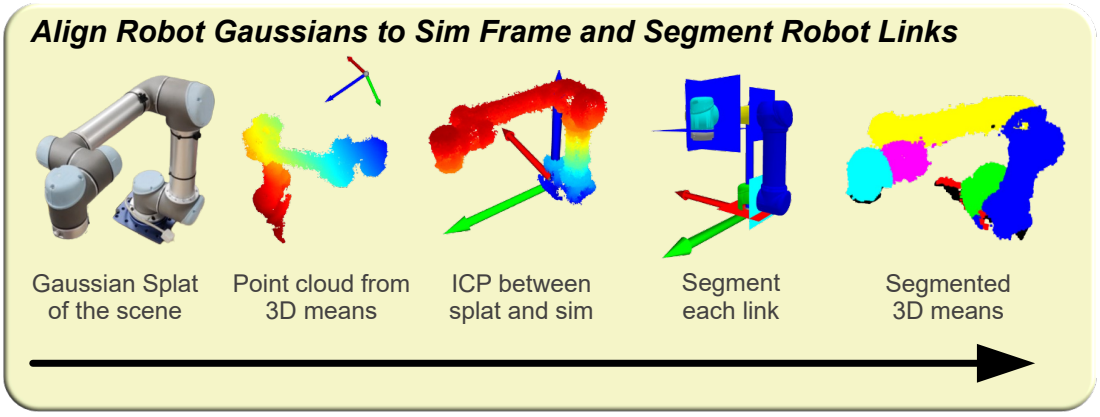

Figure 3.2: The robot is visualized in a static scene by first creating a Gaussian splat of the scene with the robot in its home position. The robot's point cloud is manually segmented and aligned with the canonical robot frame using the ICP algorithm. Each robot link is then segmented, and forward kinematics transformations are applied, enabling the rendering of the robot at arbitrary joint configurations.

Alignment of Gaussian Splat Robot Frame to the Simulator Frame

In order to combine the Gaussian Splat representation $\mathcal{S}_{real}$ with the simulator, we first manually segment out the 3D Gaussians associated with the robot. The means of these 3D Gaussians form a point cloud which is aligned with the ground truth point cloud obtained from the simulator. To achieve this, we use the Iterative Closest Point (ICP) algorithm, which produces the desired transformation $T_{\mathcal{F}_{robot}}^{\mathcal{F}_{splat}}$.

Segmentation of the Robot Links

To associate the 3D Gaussians with their respective links in $\mathcal{S}_{real}$, we leverage the ground truth bounding boxes of the robot's links, provided by its CAD model. This method allows us to isolate the 3D Gaussians corresponding to each link in the real-world scene, denoted as $\mathcal{S}_{real}^{l}$, where l refers to the l-th link of the robot.

Forward Kinematics Transformation

Once we have the 3D Gaussians for individual links and the frames aligned, we can use the robot's forward kinematics to get the robot pose at arbitrary joint angles $q_t \in s_t$, given by the simulator. In this work, we use the forward kinematics routine

from PyBullet to get the Transformation T_{fk}^{l} for link l in the robot's canonical frame $\mathcal{F}_{sim}$. The transformation of the 3D Gaussians can be calculated as :

$$T = (T_{\mathcal{F}_{robot}}^{\mathcal{F}_{splat}})^{-1} \cdot T_{fk}^{l} \cdot T_{\mathcal{F}_{robot}}^{\mathcal{F}_{splat}} \tag{3.1}$$

where $T_{\mathcal{F}_{robot}}^{\mathcal{F}_{splat}}$ is the transformation matrix to get the robot from splat frame to the simulation frame. Once the transformation for each link is calculated, we use Eq. **??** and Eq. **??** to transform the 3D Gaussians related to individual links of the robot. The robot at novel poses is then rendered by the standard Gaussian Splatting rendering framework [23].

3.4 Object Splat Models

Similar to the robot rendering, we use ICP to align each object's 3D Gaussians $\mathcal{S}_{obj}^{k}$ to its simulated ground truth point cloud. In this way, we get the transformation $T_{\mathcal{F}_{k-obj,sim}}^{\mathcal{F}_{k-obj,splat}}$, which transforms the splat in $\mathcal{S}_{obj}^{k}$ frame to simulator frame. Given the position $p_t^k \in s_t$ and orientation $R_t^k \in s_t$ can be used to calculate the transformation of object T_{fk}^{k-obj} from its original simulator frame $\mathcal{F}_{k-obj,sim}$. Using T_{fk}^{k-obj} we can get the object's 3D Gaussians in $\mathcal{S}_{real}$ frame with the transformation :

$$T = (T_{\mathcal{F}_{robot}}^{\mathcal{F}_{splat}})^{-1} \cdot T_{fk}^{k-obj} \cdot T_{\mathcal{F}_{k-obj,sim}}^{\mathcal{F}_{k-obj,splat}} \tag{3.2}$$

Once the transformation for the object is calculated, we can transform the 3D Gaussians related to the object. Then we use the Gaussian Splatting rendering framework [23] to render the object at its new position and orientation.

3.5 Articulated Object

While CAD axis-aligned bounding boxes allow straightforward segmentation of robot links, certain objects, such as parallel jaw grippers, present challenges due to their misalignment with standard axes, that is, the gripper links are not neatly segmented out by just using bounding boxes in the 3D space. To address this, we employ a ground truth K-Nearest Neighbour (KNN) classifier trained on labeled simulator

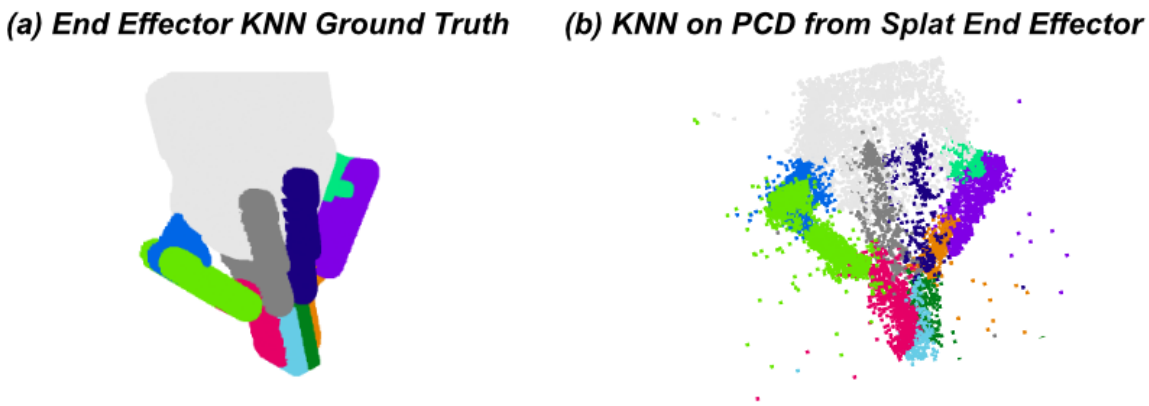

Figure 3.3: We use a KNN-based classifier for segmenting links for articulated objects like parallel jaw grippers. We train a KNN model with the ground truth point labeling from the URDF model of the end effector.

point clouds as in Fig. 3.3 (a), which enables inference of the link class for each 3D Gaussian in the aligned splat as shown in Fig. 3.3 (b).

3.6 Rendering Simulated Trajectories using SplatSim

Now that we are able to render individual rigid bodies in the scene, we can use this to represent any simulated trajectory $\tau_{\mathcal{E}}$ with photorealistic accuracy. We use these state-based transformations along with methods described in Sec. 3.3, 3.4 to get the demonstration for our policy to learn from $\tau_{\mathcal{G}} = \{(I_1^{sim}, a_1), (I_2^{sim}, a_2), \ldots, (I_T^{sim}, a_T)\}$. This data is used by policy to predict actions from the synthetically generated images.

3.7 Policy Training and Deployment

For learning from the generated demonstrations $\tau_{\mathcal{G}}$ in the simulator, we employ Diffusion Policy [9, 10], which is the state of the art for behavior cloning. Although our method significantly mitigates the vision Sim2Real gap, discrepancies between the simulated and real-world environments remain. For instance, simulated scenes lack shadows, and rigid body assumptions can lead to improper rendering of flexible components such as robot cables. To address these issues, we incorporate image

augmentations similar to [6] during policy training, which includes adding Gaussian noise, random erasing and adjusting brightness and contrast of the image. These augmentations notably enhance the robustness of the policy and improve its performance during real-world deployment.

Chapter 4

Experiments

To evaluate the effectiveness of our framework in bridging the Sim2Real gap for RGB-based manipulation tasks, we conducted extensive experiments across four real-world manipulation tasks as in Fig. 4.1. We begin by detailing the data collection process in both the simulator and real-world environments. We then compare the performance of policies trained on our synthetic data with Real2Real policies—those trained on real-world data and deployed in real-world environments. This comparison demonstrates the high fidelity of our synthetic data, showing that policies trained within our framework can be deployed to real-world tasks without fine-tuning. Additionally, we assess Sim2Sim performance by training and evaluating policies entirely within the *SplatSim*, allowing us to quantify the degradation in performance during Sim2Real transfer. Lastly, we investigate the effects of data augmentation on the transfer process and evaluate the visual fidelity of the photorealistic renderings generated by the *SplatSim* framework.

4.1 Demonstrations in the Real World and Simulation

In the real world, demonstrations for each task were manually collected by a human expert. In contrast, the simulator streamlines this process by employing privileged information-based motion planners, which automatically generate data using privi-

Task	Successful Trials (Out of 40 Trials)			Human Effort to Collect Data (hours)	
	Sim2Sim	Real2Real	Sim2Real **(SplatSim)**	Simulator	Real World
T-Push	100%	100%	90%	3.0	3.5
Pick-Up-Apple	100%	100%	95%	0.0*	3.5
Orange-On-Plate	97.5%	95%	90%	0.0*	6.0
Assembly	85%	90%	70%	0.0*	7.5
Total	95.62%	97.5%	**86.25**%	3.0	20.5

* Automated process

Table 4.1: Comparison of task success rates and data collection times across various manipulation tasks. Our policies trained solely on synthetic data achieve an 86.25% zero-shot Sim2Real performance, comparable to those trained on real-world data. By leveraging the automation capabilities of simulators, we significantly reduce the human effort required for data generation.

leged information, such as the position and orientation of each rigid body in the scene. The simulator not only reduces effort by automating resets between demonstrations when a human expert is involved but more importantly, it leverages motion planners that eliminate the need for human intervention entirely. This enables the generation of large-scale, high-quality demonstration datasets with minimal manual input. As a result, the simulator drastically reduces the time and effort required for data collection. As shown in Table 4.1, while real-world demonstration collection required about 20.5 hours, the same tasks were completed in just 3 hours in the simulator, underscoring the efficiency and scalability of our approach.

4.2 Zero-Shot Policy Deployment Results

We evaluate the zero-shot deployment of our policies across four contact-rich real-world tasks, using task success rate as the primary metric. As shown in Table 4.1, our method achieves an average success rate of 86.25% for zero-shot Sim2Real transfer, compared to 97.5% for policies trained directly on real-world data, highlighting the effectiveness of our approach. All experiments were conducted using a UR5 robot equipped with a Robotiq 2F-85 gripper and 2 Intel Realsense D455 cameras [24] with deployment on an NVIDIA RTX 3080Ti GPU for the Diffusion Policy [9].

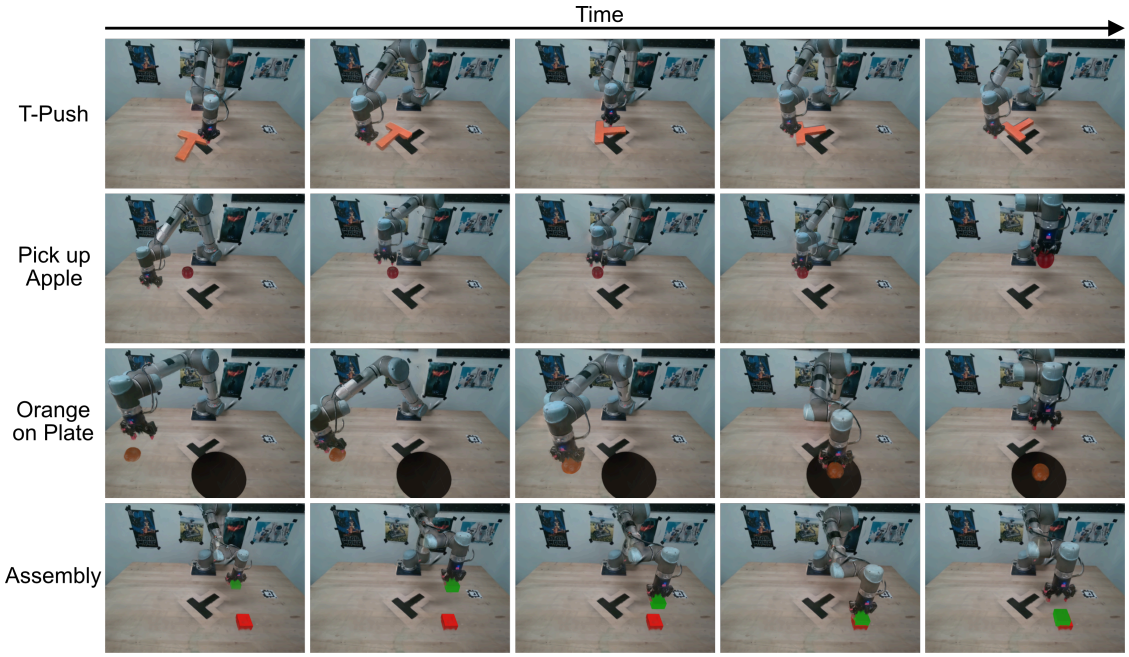

Figure 4.1: SplatSim Rollout: Renderings from our SplatSim framework across four different manipulation tasks.

T-Push Task

The T-Push task, popularized by Diffusion Policy [9], captures the dynamics of non-prehensile manipulation, which involves controlling both object motion and contact forces. For training, a human expert collected 160 demonstrations in simulation using the Gello teleoperation [46]. While testing, the robot started from a random location and achieved a 90% success rate (36/40 trials) in zero-shot Sim2Real transfer as shown in Table 4.1. This result shows the effectiveness of our framework in handling the dynamics of pushing without fine-tuning on real-world demonstrations. Additionally, the performance of our method is comparable to Real2Real (40/40) and Sim2Sim (40/40).

Pick-Up-Apple Task

The Pick-Up-Apple task involves grasping and manipulating the full pose of an object (i.e., position and orientation) in 3D. This task was designed to evaluate the robot's grasping capabilities when trained using our simulated renderings. A motion planner,

leveraging privileged state information from the simulator (accurate position and orientation of each rigid body in the scene), generated 400 demonstrations with randomized end-effector positions and orientations. During real-world trials, our policy achieved a 95% success rate (38/40 trials) in zero-shot Sim2Real transfer, as shown in Table 4.1.

Orange on Plate Task

In this task the robot has to pick up an orange and place it on a plate. In simulation, a motion planner with access to privileged information, generated 400 demonstrations. The end-effector position and initial gripper state were randomized during training. During testing, the robot always started from a home position. We achieved a 90% success rate (36/40 trials) in zero-shot Sim2Real transfer.

Assembly Task

In this task the robot has to put a cuboid block on top of another cuboid. The robot starts at the home position with the green cube already grasped and has to place it on top of the red cube. The task is particularly tough since the robot has to make a precise placement otherwise the cube will fall and will lead to a failure case. Our Sim2Real policy achieved a performance of 70% (28/40 trials) on this task, compared to 95% on Sim2Sim and 90% on Real2Real.

4.3 Quantifying Robot Renderings

We quantitatively evaluate the accuracy of rendered robot images at various joint configurations by comparing them with the real-world images. We assess the quality of the robot's renderings across 300 different robot joint angles. To measure the similarity between the rendered and real-world images, we employ two metrics commonly used in image rendering assessment: Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM). Despite the variations in joint configurations, the renderings achieve an average PSNR of 22.62 and an SSIM of 0.7845, indicating that the simulated images closely approximates the visual quality of the real-world RGB observations.

4.4 Effect of Augmentations

To quantify the impact of data augmentations on the Sim2Real performance of our policy, we conducted experiments comparing policies trained with and without augmentations. While the Diffusion-Policy performs effectively without augmentations in consistent environments (e.g., Sim2Sim or Real2Real scenarios), transferring a policy trained in simulation to the real world introduces domain shifts that necessitate additional robustness as the renderings can't capture dynamic details like changing reflections and shadows. We incorporated augmentations such as random noise addition, Color Jitter, and random erasing during training to address these shifts. These augmentations improve the performance of the policy from 21% to 86.25% across four tasks in Sec. 4.2.

Chapter 5

Conclusions

5.1 Conclusions

In this work, we tackled the Sim2Real gap for RGB-based manipulation policies by leveraging Gaussian Splatting for photorealistic rendering, integrated with existing simulators for physics-based interactions. Our framework enables zero-shot transfer of RGB policies trained in simulation to real-world environments, significantly reducing the need for real-world data collection. While advancing the state-of-the-art in photorealistic simulation and zero-shot deployment, our current system remains limited to rigid-body manipulation and controlled tabletop scenes.

Extending this framework to more dynamic and unstructured domains such as agriculture, where tasks like pruning and harvesting require complex, contact-rich interactions, poses several challenges. We outline three important directions for future work:

1. **Modeling Deformable and Organic Structures:** Real-world plants such as vines exhibit non-rigid, highly deformable behaviors that are difficult to simulate. Addressing this will require incorporating deformation-aware models, either through physics-informed approximations or by learning dynamics directly from real-world data using deformable Gaussian splatting or video-driven deformation modules.
2. **Scalable Scene Representations:** Agricultural scenes often span large out-

door environments with many objects and complex topologies. Scaling our framework to such settings demands a more efficient and hierarchical representation of splats, potentially through multi-resolution scene graphs or chunked splat loading strategies that support real-time interaction over large spatial extents.

3. **Robust Policy Learning in Unstructured Environments:** The visual variability and uncertainty in field settings (e.g., lighting changes, occlusions, moving foliage) can degrade policy performance. To address this, future work should explore reinforcement learning with heavy visual and physical domain randomization, online adaptation techniques, and closed-loop visual feedback systems to ensure robust policy execution in dynamic real-world conditions.

By addressing these challenges, we aim to extend the capabilities of our system toward scalable, real-world deployments in complex outdoor manipulation tasks such as pruning and harvesting, ultimately bridging the gap between simulation and impactful field robotics applications.

Appendix A

Extending SplatSim to Articulated and Deformable objects

A.1 Pipeline for Articulated Objects

For articulated objects such as robot grippers and drawers, we adopt a segmentation and rendering pipeline inspired by the KNN-based method described in the SplatSim framework. The steps for rendering articulated objects in novel configurations are outlined below.

A.1.1 Robot Link Segmentation Using KNN Classifier

We utilize a K-Nearest Neighbour (KNN) classifier trained on ground-truth point labels from the simulator to segment the Gaussian splats corresponding to individual links of the articulated object. This approach accounts for misaligned or non-axis-aligned bounding boxes, such as those found in parallel jaw grippers. After training, the KNN classifier infers the link class for each Gaussian in the aligned splat, ensuring accurate segmentation.

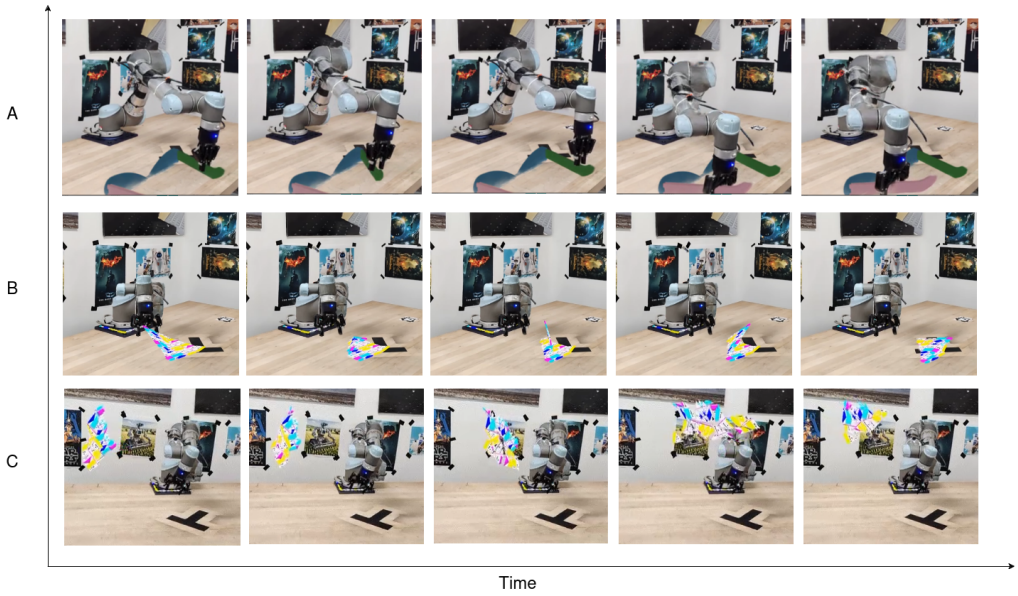

Figure A.1: The figure illustrates temporal sequences for different object manipulations. (A) Demonstrates the progression of articulated object manipulation, showcasing the movement of eyeglass handles over time. (B) Presents the time steps involved in soft object manipulation, specifically depicting the folding process of a cloth. (C) Displays the limitation of the current method of rendering clothes for extreme deformations.

A.1.2 Forward Kinematics Transformation for Articulated Links

Once the Gaussian splats are segmented, we calculate the transformations for each link using forward kinematics. The transformation matrix for each link is computed relative to the robot's base as:

$$\mathbf{T} = \mathbf{T}_{\text{base}} \cdot \mathbf{T}_{\text{link}}, \tag{A.1}$$

where $\mathbf{T}_{\text{base}}$ is the transformation of the robot base and $\mathbf{T}_{\text{link}}$ is the local transformation of the link obtained from the simulator.

A.1.3 Rendering Articulated Objects

After transforming the segmented Gaussians, we project them onto the image plane using Gaussian Splatting. This pipeline ensures photorealistic renderings of articulated objects at novel joint configurations, maintaining high fidelity to the simulated scene.

A.2 Segmentation of Cloth Gaussians into Mesh Faces

To accurately associate Gaussian splats with the corresponding mesh faces of a cloth object, we segment the point cloud based on the mesh structure. This segmentation allows us to maintain a one-to-one correspondence between the simulation's mesh-based representation of the cloth and its Gaussian splats in the rendered space. The steps for segmentation are detailed below.

A.2.1 2D Projection of Mesh Vertices and Points

Given the 3D nature of both the mesh and the Gaussian splats, we project the vertices of the mesh and the points in the Gaussian splats onto a 2D plane. This is achieved by ignoring the z-axis coordinates:

$$(x, y, z) \rightarrow (x, y). \tag{A.2}$$

This simplification reduces the problem of associating points with mesh faces to a 2D problem, making computations more efficient.

A.2.2 Point-in-Triangle Test Using Barycentric Coordinates

For each projected point, we determine its association with a specific triangle (mesh face) using the point-in-triangle test. This is performed by computing the signed areas of the triangles formed by the point and the vertices of each triangle. Let the vertices of a triangle be (x_A, y_A), (x_B, y_B), and (x_C, y_C). The total area of the triangle

is given by:

$$\text{Area}_{ABC} = \frac{1}{2} \left| x_A(y_B - y_C) + x_B(y_C - y_A) + x_C(y_A - y_B) \right|. \tag{A.3}$$

For a point (x_P, y_P) to lie inside the triangle, the sum of the areas of sub-triangles formed by the point and the vertices of the triangle must equal the total area:

$$\text{Area}_{PAB} + \text{Area}_{PBC} + \text{Area}_{PCA} = \text{Area}_{ABC}. \tag{A.4}$$

Assigning Labels to Points

If a point is determined to lie inside a triangle, it is assigned the label of that triangle. This mapping is stored for all points, resulting in a complete segmentation of the Gaussian splats into mesh faces.

A.3 Deformation of Cloth with Rigid Body Approximation

To model cloth deformation, we assume that each triangle (mesh face) behaves as a rigid body. Although this is an approximation, it allows us to compute transformations for each triangle independently, facilitating the rendering of deformed cloth in novel configurations.

A.3.1 Retrieving Vertex Data

From the simulation, we retrieve the current vertex positions $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ for each triangle and their initial positions $\mathbf{v}_1^0, \mathbf{v}_2^0, \mathbf{v}_3^0$.

A.3.2 Centroid Calculation

The centroid of a triangle is calculated as the average of its three vertices. For the current and initial configurations, the centroids are given by:

$$\mathbf{c} = \frac{\mathbf{v}_1 + \mathbf{v}_2 + \mathbf{v}_3}{3}, \quad \mathbf{c}^0 = \frac{\mathbf{v}_1^0 + \mathbf{v}_2^0 + \mathbf{v}_3^0}{3}. \tag{A.5}$$

A.3.3 Aligning Faces Using SVD

To align each triangle's initial configuration with its current state, we compute the optimal rotation matrix $\mathbf{R}$ using Singular Value Decomposition (SVD). The covariance matrix $\mathbf{H}$ is calculated as:

$$\mathbf{H} = \sum_{i=1}^{3} (\mathbf{v}_i^0 - \mathbf{c}^0)(\mathbf{v}_i - \mathbf{c})^\top. \tag{A.6}$$

SVD is performed on $\mathbf{H}$:

$$\mathbf{H} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top, \tag{A.7}$$

and the rotation matrix $\mathbf{R}$ is obtained as:

$$\mathbf{R} = \mathbf{V}\mathbf{U}^\top. \tag{A.8}$$

A.3.4 Calculating Translation

The translation vector $\mathbf{t}$ is computed as the difference between the centroids after aligning the triangle:

$$\mathbf{t} = \mathbf{c} - \mathbf{R}\mathbf{c}^0. \tag{A.9}$$

A.3.5 Constructing Transformation Matrix

The transformation matrix for each triangle is represented in homogeneous coordinates as:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}. \tag{A.10}$$

A.4 Limitations of the Current Approach

We are assuming each face of the mesh as a rigid body. However, this is not true, as the triangle itself deforms in the simulator. Incorporating those cases is important. The below video shows this. The gap between the triangles is because we are not scaling the gaussians properly according to the deformation of the triangle.

A.5 Experiments

Our experiments primarily focused on qualitative assessments due to the inherent challenges in quantifying the performance of cloth simulation without a real-world counterpart. The lack of a physical cloth setup in our experimental environment limited our ability to conduct direct comparisons or measure precise metrics. However, we were able to evaluate the visual fidelity of our cloth rendering using the Peak Signal-to-Noise Ratio (PSNR) of the Gaussian splatting representation compared to the ground truth mesh-based rendering. The PSNR value of 25.12 demonstrates the high quality of our cloth rendering. Detailed visual results are presented in Figure 1, which showcases the temporal sequences of articulated objects and cloth renderings.

Appendix B

Applying Scene Modelling to the Field Setting

B.1 Motivation

Agricultural robotics presents unique challenges that distinguish it from typical indoor robotic manipulation settings. Unlike controlled environments, field deployments are affected by factors such as seasonality, changing weather conditions, unstructured backgrounds, and limited infrastructure. One of the most significant bottlenecks in developing robotic systems for agriculture is the difficulty of collecting large-scale, high-quality real-world data. Robots often need to be transported to remote fields, setup times are high, and hardware is prone to failure due to environmental exposure. These challenges make scalable and repeatable data collection for robot learning extremely difficult.

To address these issues, we explore the use of *Gaussian Splatting* as a tool for modeling complex agricultural scenes with high photorealism and geometric accuracy. Gaussian Splats provide a dense and editable 3D representation that can be reconstructed from a short RGB video captured in the field. Once reconstructed, these splats serve as a visual twin of the real environment, enabling offline planning and simulation without requiring repeated trips to the field.

We specifically target the problem of *grapevine pruning*, a task that is vital for

crop health and yield, yet challenging to automate due to the intricacies involved in identifying appropriate pruning points. In our approach, we first scan the grapevine row to obtain a Gaussian Splat model of the scene. This splat is then annotated by expert farmers who mark the pruning locations directly in the 3D space. Once labeled, the robot uses this annotated splat to plan and execute pruning actions in the real world. This system significantly reduces the need for repeated manual intervention, leverages human expertise efficiently, and offers a scalable solution to a traditionally labor-intensive task.

B.2 System Overview

Our proposed field deployment pipeline begins with capturing a raw RGB video of the scene, which is then used to generate a Gaussian Splat representation. Once this splat is available, the system proceeds through three main components: scaling, localization, and motion planning, as outlined in Fig. B.1.

1. Splat Generation and Scaling: The initial step involves reconstructing a 3D Gaussian Splat of the agricultural scene from a multi-view RGB sequence. However, since the reconstruction is up to scale, we need to calibrate the scene to real-world dimensions. This is achieved by including a scaling object—typically a cube or cuboid with known dimensions—in the captured sequence. The dimensions of the object in the reconstructed splat are measured and used to compute a global scale factor, which is then applied to all 3D Gaussians in the scene. This step ensures that any positions or distances measured within the splat are consistent with the robot's operational frame.

2. Localization of the Robot in the Splat Frame: For the robot to interact meaningfully with the Gaussian Splat, it is essential to localize the robot's base within the splat's coordinate frame. This is typically done by performing a registration step between the known CAD model or a depth scan of the robot and its observed splat reconstruction. Alternatively, localization can be achieved by visual servoing, where the robot adjusts its position until its camera view matches a rendered view from the splat. This results in the transformation H_c^s, which maps the robot's camera frame to the splat frame.

3. Cartesian Motion Planning: Once the splat is scaled and the robot is

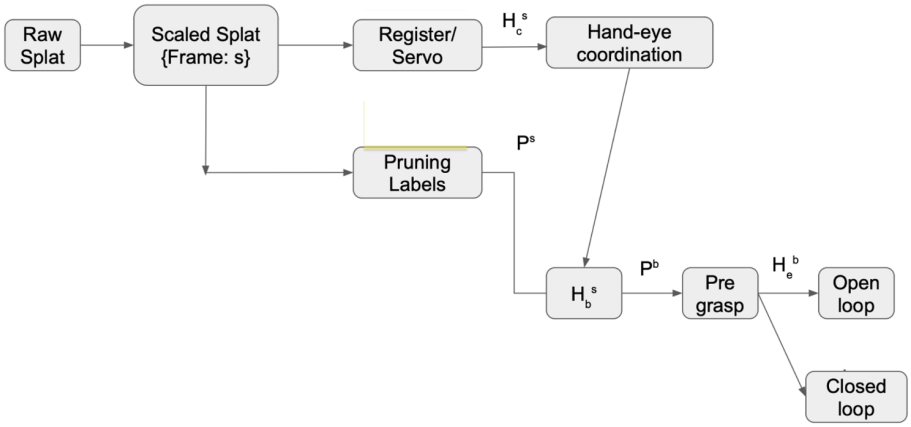

Figure B.1: Overview of the field deployment pipeline. A raw Gaussian Splat is first scaled using a known calibration object. The robot is then localized in the splat frame via servoing or registration. Once localized, expert-annotated pruning points in the splat can be transformed into the robot frame, enabling planning and execution of pruning actions using simple Cartesian motion. The system supports both open-loop and closed-loop execution.

localized within it, we can leverage the annotated pruning points within the splat to guide manipulation. These labeled points are typically provided by an expert who marks pruning locations directly in the 3D splat. Given a target point, we can use the transformation from the splat frame to the robot base frame to compute the goal in the robot's coordinate system. A simple Cartesian planner can then be used to generate a trajectory from the robot's current end-effector pose to the goal. This motion can either be executed in an open-loop manner or refined via closed-loop visual feedback.

B.3 Robot Localization via Visual Servoing in Gaussian Splat Space

A critical requirement for executing any manipulation in the field is knowing the robot's pose relative to the environment model—in our case, the Gaussian Splat. Traditional localization techniques rely on fiducial markers or external sensors, which can be impractical in outdoor agricultural settings. Instead, we propose a novel form

of localization using visual servoing within the Gaussian Splat itself.

In this approach, the robot captures a single RGB image from its onboard camera. To estimate the camera pose relative to the splat, we render views from the Gaussian Splat by simulating a virtual camera and adjusting its pose until the rendered image visually matches the robot's captured image. This optimization process aligns the real camera view with the virtual splat rendering, effectively yielding the transformation H_c^s from the camera frame to the splat frame.

Unlike traditional pose-based visual servoing that moves the robot to match a known goal image, we invert the process: we move a virtual camera in the splat to match the real-world observation. This formulation avoids any need for hand-crafted features, is agnostic to scene content, and can work with natural environments such as grapevines under varying lighting conditions.

This virtual camera servoing is inspired by direct image-based visual servoing (IBVS) and flow-guided methods like DFVS [17], but does not require scene-specific training. Once the best-fit pose is found in the splat frame, it can be composed with known robot kinematics to transform pruning targets or waypoints from the splat space into the robot's base frame. This allows us to bridge the perception and planning modules without requiring external localization infrastructure.

B.4 Pregrasp Planning and Execution

Once the robot is localized within the Gaussian Splat using visual servoing, the next step is to plan and execute a motion to interact with annotated points in the environment. In our application of grapevine pruning, these points are labeled by expert farmers directly within the 3D splat. Each pruning location is defined in the splat frame $\{s\}$ as a 6DoF pose, including both position and orientation, denoted as P^s.

Given the transformation H_c^s estimated during visual servoing, and the known hand-eye calibration between the camera and the robot's end-effector, we compute the corresponding pose in the robot base frame $\{b\}$. Specifically, we use:

$$P^b = H_b^c \cdot (H_c^s)^{-1} \cdot P^s$$

This yields the desired pruning point in the robot's coordinate system. From this target pose, a Cartesian motion plan is generated to guide the end-effector from its current pose to the pruning location. In our implementation, this motion can be executed in one of two modes:

- **Open-Loop Execution:** If the localization from visual servoing is precise and the splat alignment is reliable, the robot can directly follow a planned trajectory to the target pose without additional feedback. This approach is simple and effective in well-observed scenes.
- **Closed-Loop Visual Feedback:** In cases where uncertainty in localization or scene alignment exists (e.g., due to occlusions or changes in lighting), we enable closed-loop control. Here, the robot uses real-time camera feedback to minimize the visual discrepancy between the current view and a rendered goal view from the splat. This helps correct small pose errors during motion and ensures accurate placement at the pruning point.

This pregrasp execution step effectively bridges the annotated splat model with low-level robot control, enabling precise and safe interaction with delicate plant structures in outdoor environments.

B.4.1 Discussion and Conclusion

This appendix outlines a practical extension of the SplatSim framework to challenging agricultural field settings, with a specific focus on grapevine pruning. By leveraging the photorealistic and editable nature of Gaussian Splatting, we construct a visual twin of the environment that can be labeled by experts and used for downstream robot planning and execution. Our system circumvents the need for repeated real-world data collection by enabling expert annotations directly in the splat and using visual servoing for robot localization.

A key feature of this pipeline is the inversion of traditional visual servoing: rather than controlling the robot to reach a rendered target image, we move a virtual camera within the splat until it matches the live image from the robot's camera. This yields a transformation between the robot and the scene, which can be composed with annotated points for precise 6DoF goal specification.

Despite the system's promise, several open challenges remain. The quality of the Gaussian Splat reconstructions can degrade under difficult lighting conditions or with highly deformable foliage. Furthermore, achieving high-precision localization still depends on the visual richness and coverage of the captured views. In dynamic environments—such as those affected by wind or seasonal changes—the splat may need to be updated more frequently to remain useful.

Future directions include integrating deformable or time-varying splats, supporting multiple robot-camera configurations, and fusing visual servoing with SLAM-based localization to improve robustness. Overall, this pipeline demonstrates the feasibility of applying simulation-informed techniques in real-world agricultural settings with minimal infrastructure and manual effort.

Bibliography

[1] Jad Abou-Chakra, Krishan Rana, Feras Dayoub, and Niko Suenderhauf. Physically embodied gaussian splatting: A realtime correctable world model for robotics. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=AEq0onGrN2. 2.2

[2] Ananye Agarwal, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Legged locomotion in challenging terrains using egocentric vision. *CoRL*, 2022. 1, 2.1

[3] C.E. Aguero, N. Koenig, I. Chen, H. Boyer, S. Peters, J. Hsu, B. Gerkey, S. Paepcke, J.L. Rivero, J. Manzo, E. Krotkov, and G. Pratt. Inside the virtual robotics challenge: Simulating real-time robotic disaster response. *Automation Science and Engineering, IEEE Transactions on*, 12(2):494–506, April 2015. ISSN 1545-5955. doi: 10.1109/TASE.2014.2368997. 2.1

[4] Jeongmin Bae, Seoha Kim, Youngsik Yun, Hahyun Lee, Gun Bang, and Youngjung Uh. Per-gaussian embedding-based deformation for deformable 3d gaussian splatting. In *European Conference on Computer Vision (ECCV)*, 2024. 2.2

[5] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. *ICCV*, 2021. 2.2

[6] Arunkumar Byravan, Jan Humplik, Leonard Hasenclever, Arthur Brussee, Francesco Nori, Tuomas Haarnoja, Ben Moran, Steven Bohez, Fereshteh Sadeghi, Bojan Vujatovic, and Nicolas Heess. Nerf2real: Sim2real transfer of vision-guided bipedal motion skills using neural radiance fields. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9362–9369, 2023. doi: 10.1109/ICRA48891.2023.10161544. 3.7

[7] Yurui Chen, Chun Gu, Junzhe Jiang, Xiatian Zhu, and Li Zhang. Periodic vibration gaussian: Dynamic urban scene reconstruction and real-time rendering. *arXiv:2311.18561*, 2023. 2.2

[8] Xuxin Cheng, Kexin Shi, Ananye Agarwal, and Deepak Pathak. Extreme parkour with legged robots. In *2024 IEEE International Conference on Robotics and*

Automation (ICRA), pages 11443–11450, 2024. doi: 10.1109/ICRA57147.2024.10610200. 1, 2.1

[9] 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. (document), 3.1, 3.7, 4.2, 4.2

[10] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2024. 3.7

[11] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021. 2.1

[12] Bardienus P. Duisterhof, Zhao Mandi, Yunchao Yao, Jia-Wei Liu, Jenny Seidenschwarz, Mike Zheng Shou, Deva Ramanan, Shuran Song, Stan Birchfield, Bowen Wen, and Jeffrey Ichnowski. Deformgs: Scene flow in highly deformable scenes for deformable object manipulation. In *The 16th International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2024. 2.2

[13] Ben Eisner, Harry Zhang, and David Held. FlowBot3D: Learning 3D Articulation Flow to Manipulate Articulated Objects. In *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, June 2022. doi: 10.15607/RSS.2022.XVIII.018. 2.1

[14] Zipeng Fu, Xuxin Cheng, and Deepak Pathak. Deep whole-body control: Learning a unified policy for manipulation and locomotion. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 138–149. PMLR, 14–18 Dec 2023. URL https://proceedings.mlr.press/v205/fu23a.html. 2.1

[15] Jiayuan Gu, Fanbo Xiang, Xuanlin Li, Zhan Ling, Xiqiang Liu, Tongzhou Mu, Yihe Tang, Stone Tao, Xinyue Wei, Yunchao Yao, Xiaodi Yuan, Pengwei Xie, Zhiao Huang, Rui Chen, and Hao Su. Maniskill2: A unified benchmark for generalizable manipulation skills. In *International Conference on Learning Representations*, 2023. 2.1

[16] Ankur Handa, Arthur Allshire, Viktor Makoviychuk, Aleksei Petrenko, Ritvik Singh, Jingzhou Liu, Denys Makoviichuk, Karl Van Wyk, Alexander Zhurkevich, Balakumar Sundaralingam, and Yashraj Narang. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5977–5984, 2023. doi:

10.1109/ICRA48891.2023.10160216. 1, 2.1

[17] YVS Harish, Harit Pandya, Ayush Gaud, Shreya Terupally, Sai Shankar, and K Madhava Krishna. Dfvs: Deep flow guided scene agnostic image based visual servoing. *arXiv preprint arXiv:2003.03766*, 2020. B.3

[18] David Hoeller, Nikita Rudin, Dhionis Sako, and Marco Hutter. Anymal parkour: Learning agile navigation for quadrupedal robots, 2023. URL https://arxiv.org/abs/2306.14874. 2.1

[19] Binghao Huang, Yuanpei Chen, Tianyu Wang, Yuzhe Qin, Yaodong Yang, Nikolay Atanasov, and Xiaolong Wang. Dynamic handover: Throw and catch with bimanual hands. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=dgwvY3H8PAS. 2.1

[20] Stephen James, Andrew J. Davison, and Edward Johns. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 334–343. PMLR, 13–15 Nov 2017. URL https://proceedings.mlr.press/v78/james17a.html. 2.1

[21] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12619–12629, 2019. doi: 10.1109/CVPR.2019.01291. 2.1

[22] Mazeyu Ji, Ri-Zhao Qiu, Xueyan Zou, and Xiaolong Wang. Graspsplats: Efficient manipulation with 3d feature splatting. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=pPhTsonbXq. 2.2

[23] 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/. (document), 1.1, 1, 2.2, 3.3, 3.4

[24] Leonid Keselman, John Iselin Woodfill, Anders Grunnet-Jepsen, and Achintya Bhowmik. Intel(r) realsense(tm) stereoscopic depth cameras. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1267–1276, 2017. doi: 10.1109/CVPRW.2017.167. 4.2

[25] Ashish Kumar, Zhongyu Li, Jun Zeng, Deepak Pathak, Koushil Sreenath, and Jitendra Malik. Adapting rapid motor adaptation for bipedal robots. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1161–1168, 2022. doi: 10.1109/IROS47612.2022.9981091. 2.1

[26] Antonio Loquercio, Elia Kaufmann, René Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Deep drone racing: From simulation to reality with domain randomization. *IEEE Transactions on Robotics*, 2019. doi: 10.1109/TRO.2019.2942989. 2.1

[27] Haozhe Lou, Yurong Liu, Yike Pan, Yiran Geng, Jianteng Chen, Wenlong Ma, Chenglong Li, Lin Wang, Hengzhen Feng, Lu Shi, Liyi Luo, and Yongliang Shi. Robo-gs: A physics consistent spatial-temporal model for robotic arm with hybrid representation, 2024. URL https://arxiv.org/abs/2408.14873. 2.2

[28] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, 2024. 2.2

[29] Tyler Ga Wei Lum, Martin Matak, Viktor Makoviychuk, Ankur Handa, Arthur Allshire, Tucker Hermans, Nathan D. Ratliff, and Karl Van Wyk. Dextrah-g: Pixels-to-action dexterous arm-hand grasping with geometric fabrics, 2024. URL https://arxiv.org/abs/2407.02274. 1, 2.1

[30] Jan Matas, Stephen James, and Andrew J. Davison. Sim-to-real reinforcement learning for deformable object manipulation. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 734–743. PMLR, 29–31 Oct 2018. URL https://proceedings.mlr.press/v87/matas18a.html. 2.1

[31] 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. 2.2

[32] Mayank Mittal, Calvin Yu, Qinxi Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Ritvik Singh, Yunrong Guo, Hammad Mazhar, Ajay Mandlekar, Buck Babich, Gavriel State, Marco Hutter, and Animesh Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023. doi: 10.1109/LRA.2023.3270034. 2.1

[33] Tongzhou Mu, Zhan Ling, Fanbo Xiang, Derek Yang, Xuanlin Li, Stone Tao, Zhiao Huang, Zhiwei Jia, and Hao Su. Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. In Joaquin Vanschoren and Sai-Kit Yeung, editors, *NeurIPS Datasets and Benchmarks*, 2021. URL http://dblp.uni-trier.de/db/conf/nips/neurips2021db.html#MuLXYLLTHJ021. 2.1

[34] 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. 2.2

[35] Haozhi Qi, Ashish Kumar, Roberto Calandra, Yi Ma, and Jitendra Malik. In-hand object rotation via rapid motor adaptation. In *6th Annual Conference on Robot Learning*, 2022. URL https://openreview.net/forum?id=Xux9gSS7WE0. 1, 2.1

[36] Alex Quach, Makram Chahine, Alexander Amini, Ramin Hasani, and Daniela Rus. Gaussian splatting to real world flight navigation transfer with liquid networks. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=ubq7Co6Cbv. 2.2

[37] Ilija Radosavovic, Tete Xiao, Bike Zhang, Trevor Darrell, Jitendra Malik, and Koushil Sreenath. Learning humanoid locomotion with transformers. *arXiv:2303.03381*, 2023. 2.1

[38] Kanishka Rao, Chris Harris, Alex Irpan, Sergey Levine, Julian Ibarz, and Mohi Khansari. Rl-cyclegan: Reinforcement learning aware simulation-to-real. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11154–11163, 2020. doi: 10.1109/CVPR42600.2020.01117. 2.1

[39] Olaolu Shorinwa, Johnathan Tucker, Aliyah Smith, Aiden Swann, Timothy Chen, Roya Firoozi, Monroe David Kennedy, and Mac Schwager. Splat-MOVER: Multi-stage, open-vocabulary robotic manipulation via editable gaussian splatting. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=8XFT1PatHy. 2.2

[40] Entong Su, Chengzhe Jia, Yuzhe Qin, Wenxuan Zhou, Annabella Macaluso, Binghao Huang, and Xiaolong Wang. Sim2real manipulation on unknown objects with tactile-based reinforcement learning, 2024. URL https://arxiv.org/abs/2403.12170. 2.1

[41] Stone Tao, Fanbo Xiang, Arth Shukla, Yuzhe Qin, Xander Hinrichsen, Xiaodi Yuan, Chen Bao, Xinsong Lin, Yulin Liu, Tse kai Chan, Yuan Gao, Xuanlin Li, Tongzhou Mu, Nan Xiao, Arnav Gurha, Zhiao Huang, Roberto Calandra, Rui Chen, Shan Luo, and Hao Su. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai, 2024. URL https://arxiv.org/abs/2410.00425. 2.1

[42] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017. doi: 10.1109/IROS.2017.8202133. 2.1

[43] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS*, pages 5026–5033. IEEE, 2012. ISBN 978-

1-4673-1737-5. URL http://dblp.uni-trier.de/db/conf/iros/iros2012.html#TodorovET12. 2.1

[44] Marcel Torne Villasevil, Anthony Simeonov, Zechu Li, April Chan, Tao Chen, Abhishek Gupta, and Pulkit Agrawal. Reconciling Reality through Simulation: A Real-To-Sim-to-Real Approach for Robust Manipulation. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.015. 2.1

[45] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20310–20320, June 2024. 2.2

[46] Philipp Wu, Yide Shentu, Zhongke Yi, Xingyu Lin, and Pieter Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators, 2023. (document), 3.1, 4.2

[47] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. *arXiv preprint arXiv:2311.12198*, 2023. 2.2

[48] Zhao-Heng Yin, Binghao Huang, Yuzhe Qin, Qifeng Chen, and Xiaolong Wang. Rotating without seeing: Towards in-hand dexterity through touch. In Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and Jingjin Yu, editors, *Robotics: Science and Systems*, 2023. ISBN 978-0-9923747-9-2. URL http://dblp.uni-trier.de/db/conf/rss/rss2023.html#YinHQCW23. 1, 2.1

[49] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021. 2.2

[50] Haonan Yu. On sim2real transfer in robotics. https://www.haonanyu.blog/post/sim2real/, 2024. Blog post. 1

[51] Ying Yuan, Haichuan Che, Yuzhe Qin, Binghao Huang, Zhao-Heng Yin, Kang-Won Lee, Yi Wu, Soo-Chul Lim, and Xiaolong Wang. Robot synesthesia: In-hand manipulation with visuotactile sensing. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6558–6565, 2024. doi: 10.1109/ICRA57147.2024.10610532. 1, 2.1

[52] Zhecheng Yuan, Tianming Wei, Shuiqi Cheng, Gu Zhang, Yuanpei Chen, and Huazhe Xu. Learning to manipulate anywhere: A visual generalizable framework for reinforcement learning. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=jart4nhCQr. 2.1

[53] Harry Zhang, Ben Eisner, and David Held. Flowbot++: Learning generalized articulated objects manipulation via articulation projection. In *7th Annual*

Conference on Robot Learning, 2023. 2.1

[54] Wenxuan Zhou, Bowen Jiang, Fan Yang, Chris Paxton, and David Held. Hacman: Learning hybrid actor-critic maps for 6d non-prehensile manipulation. In *7th Annual Conference on Robot Learning*, 2023. 2.1

[55] Ziwen Zhuang, Zipeng Fu, Jianren Wang, Christopher Atkeson, Sören Schwertfeger, Chelsea Finn, and Hang Zhao. Robot parkour learning. In *Conference on Robot Learning (CoRL)*, 2023. 2.1