Towards Dexterous Robotic Manipulation by Imitating Experts

Yulong Li

CMU-RI-TR-25-29
May 2025

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

Thesis Committee:

Prof. Deepak Pathak, Carnegie Mellon University *(Chair)*
Prof. Max Simchowitz, Carnegie Mellon University
Kenneth Shaw, Carnegie Mellon University

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

Abstract

Imitation learning offers a scalable path for transferring complex manipulation skills from expert demonstrators to robots. However, its success hinges on capturing high-quality demonstrations and effectively transferring them to robot policies, especially in contact-rich or dynamically changing environments. This thesis explores how imitation learning, when paired with teleoperation and classical solvers can be used to teach robots dexterous manipulation skills across a range of real-world scenarios.

We first present BiDex, a bimanual teleoperation system for collecting rich demonstrations of human dexterity. By learning from this data via behavior cloning, we enable visuomotor policies that generalize to tasks such as hand-offs, tool use, and fine-grained object manipulation. Building on this, we introduce FACTR, a teleoperation system enhanced with *force feedback*, enabling contact-rich behaviors with out-of-distribution generalization. We show that imitation learning in this setting benefits from access to both force and visual modalities, leading to policies that are compliant and robust.

Beyond human experts, we demonstrate that imitation learning is also effective in learning from classical methods as experts. We show that by combining behavior cloning with teacher-student fine-tuning, we enable low-latency motion generation in cluttered or dynamic environments. Finally, we explore how to leverage 3D scene representations with Gaussian Splatting to improve view robustness of the behavior cloning policies. Together, these results showcase how imitation learning unlocks dexterous robotic manipulation skills when scaffolded with the right human interfaces, training strategies, and perception tools.

Acknowledgments

I would like to thank my advisor, Deepak, whose research vision and ambition steered me toward the right problems, and whose insights pointed me to the right methods. He is an outstanding writer and presenter—skills I continue to learn from and aspire to. I'm deeply grateful to Max. Every conversation with him sharpens my thinking and deepens my appreciation for research. His intellectual sophistication and clarity of thought are qualities I will strive to achieve.

Thanks to Kenny—though not much older than me, he has always looked out for me in the lab and lead me into true robotics research. I'm also thankful to Jason for his energy, passion, and deep knowledge of many areas of robotics that I'm still learning from. To Mohan, for first introducing me to behavior cloning, and to Russell, for our thoughtful discussions on real-world reinforcement learning during late-night teleoperation sessions. I'm grateful to Jim for leading the exciting project on Deep Reactive Policy.

Although I didn't have the chance to work directly with everyone in the lab, I greatly appreciate the time and conversations we shared. I extend special thanks to Alex Li and Mihir for engaging discussions on generative models and large-scale learning, even if they remain somewhat skeptical of robot learning, and to Alex Kirchmeyer for insightful conversations on architecture design. I am grateful to Unnat and Tal for generously sharing their experience and expertise, from which I learned immensely. I've truly enjoyed the time I've spent with Tony, Hengkai, Andrew, Lili, Shagun, and Jayesh. Late nights in the lab were far more joyful because of them.

I want to pay my deepest respects to my grandparents, who sadly passed away just months before my defense. I wish I had one more chance to hear my grandmother's stories or play another game of Xiangqi with my grandfather. I'm endlessly grateful to my parents. I've finally converted by them and started running, running my way to Boston. And last but not least, thank you to Sheqi. Even though she knows little and cares even less about robotics, she has been my best research partner.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Robotic manipulation has long been considered a cornerstone for deploying autonomous agents in real-world environments. Yet, teaching robots to perform complex, contact-rich tasks remains an enduring challenge. Dexterous skills—such as grasping delicate objects, coordinating bimanual actions, or adapting to unstructured environments—demand control policies that are both reactive and generalizable. Imitation learning (IL), which leverages expert demonstrations to train policies, offers a promising paradigm for such tasks [72, 18]. Despite its promise, the success of imitation learning hinges on several key factors: (1) how demonstrations are collected, (2) how to design learning algorithms to achieve more robust and generalizable policies.

Teleoperation plays a central role in addressing the first challenge, as it enables rich, naturalistic demonstrations without requiring kinesthetic teaching [128, 65]. However, effective teleoperation is nontrivial for high-DOF systems like bimanual robot arms and dexterous hands, requiring careful consideration of usability, accuracy, and cost [34, 123].

Furthermore, while visual input is often sufficient for simple grasping or pick-and-place tasks [22, 121], contact-rich manipulation often requires incorporating tactile or force feedback [75, 37, 68]. Yet, naively fusing visual and force signals can lead to brittle policies that overfit to a single modality, motivating curriculum-based learning strategies.

Even beyond manipulation, many robotics applications demand motion policies that can respond to dynamic obstacles and changing goals in real time. Here, we extend the idea of imitation learning by shifting the expert from a human demonstrator to a classical planner. Sampling-based and optimization-based planners [46, 58, 115, 145] provide high-quality, globally consistent solutions but rely on privileged information, such as full environment geometry, and are often too slow for real-time deployment. To address these limitations, hybrid approaches [29, 30] train neural networks on planner-generated trajectories. However, these methods often require dense scene representations or suffer from limited generalization. We explore whether behavior cloning from such planners—augmented with architectural priors and teacher-student refinement—can yield policies that retain the global awareness of planners while achieving closed-loop reactivity and real-time performance.

Finally, perception constraints in real-world systems—such as the availability of only sparse RGB views—can significantly hinder generalization that requires 3D scene understanding. Recent progress in 3D Gaussian Splatting [47] and semantic 3D representations [109, 48] provides a promising approach to reconstructing 3D scene representations,. However, most methods still require dense camera setups and offline

reconstruction [69], which limits deployment in dynamic or cluttered scenes.

In this thesis, we investigate how imitation learning can scale to dexterous robotic manipulation by addressing core challenges in data collection, multimodal learning, perception, and policy generalization. We explore how to effectively collect expert demonstrations—either from humans or planners—and how to leverage these demonstrations to train robust, reactive, and generalizable policies for contact-rich tasks in the real world.

We approach these questions through four complementary systems that pair imitation learning with human interfaces, reactive planning, and 3D perception. The thesis is organized as follows:

- Chapter 2 presents BiDex, a bimanual teleoperation platform for learning dexterous visuo-motor policies from human demonstrations.
- Chapter 3 introduces FACTR, a force-augmented system for learning compliant, contact-rich manipulation policies.
- Chapter 4 explores imitation learning for reactive motion generation under dynamic conditions.
- Chapter 5 investigates dynamic Gaussian Splatting as a tool for learning from RGB observations.
- Chapter 6 concludes with a summary of findings and future directions beyond imitation learning in robotics.

Chapter 2

BiDex: Learning Bimanual Dexterous Skills with Human Demonstrations

General-purpose robots in human environments will need to perform a wide variety of challenging manipulation tasks. This ranges from intricate movements like screwing in small objects, to cutting vegetables, to operating tools to being able to move large objects like furniture. These are tasks built around humans, and correspond to activities that people can perform. The versatility of human hands in particular is essential for finer-grained tasks ranging from writing and creating art to typing on keyboards. Hence one approach to building such versatile robot systems is to use a hardware form factor that resembles humans with two arms each equipped with a dexterous multi-fingered hands.

With the advent of data-driven machine learning methods and low-cost hardware there has been renewed interest in humanoids and dexterous hands. There is great promise for machine learning approaches to enable effective autonomous control for high-dimensional robot systems using large amounts of data [60, 3, 40, 56, 45]. A key question remains: how do we collect high-quality expert data for bimanual robots? Such a data collection system must be low-cost, easy to setup and use, low-latency and most importantly accurate enough. It should be effortless for the teleoperator to collect high quality data of robots performing complex tasks of interest to train robot policies.

To address this problem, VR headsets have become increasingly prevalent due to their easy-to-use internal body tracking systems [24, 83]. However we find that wrist tracking is often jittery and the finger tracking is inaccurate. To mitigate this, SteamVR [118] uses LiDAR which provides less noisy estimates, but requires external tracking devices which doesn't allow for data collection for mobile robot setups. For even higher fidelity readings, there is work that uses motion capture and reflective marker-based approaches such as Vicon or Optitrack [120] but they are extremely expensive and difficult to setup. An aspect of motion capture technology that has been used in robot learning in recent years are wearable gloves [123, 105, 73, 106] for hand tracking which records human fingertip position using EMF sensors. We use the accuracte Manus Meta glove as part of our system. [74]

For arm tracking, researchers in the robotics community have been recently using joint-level teleoperation for 2-finger grippers [141, 128]. They find that a low-cost 3D printed scaled teacher arm model that has the same kinematic link structure of a large robot arm can be used for accurate and effective teleoperation. These methods only provide one DOF of finger tracking instead of the twenty two plus DOF of the human

Figure 2.1: **Bimanual Dexterity**: BiDex can effortlessly teleoperate various complex tasks including pouring, scooping, hammering, chopstick picking, hanger picking, picking up basket, drilling, plate pickup and pot picking to train high-quality behavior cloning policies with over 50 degrees of freedom. Our teleoperation system and two LEAP hands [105] costs around $12k in total and is readily reproducible by academic labs.

hand. Our key insight is to develop a system that combines this joint-based arm tracking along with a Mocap fingertip glove to achieve accurate low-cost teleoperation of an arm and hand system.

Our contribution is BiDex , a system for dexterous low-cost teleoperation system for bimanual hands and arms in-the-wild in any environment. To operate, the user wears two motion capture gloves and moves naturally to complete everyday dexterous tasks. The gloves captures accurate finger tracking to map motions naturally to the robot hands. The GELLO [128] inspired arm tracking accurately tracks the human wrist position and joint angles for the robot arm. The data collected by the system can be used to train effective policies using imitation learning, after which the bimanual robot system can perform the task autonomously. BiDex costs around $6k for a pair of Manus gloves and few hundred dollars for the arm teleoperation which works for many existing robot arms. Even including the robot arms ($8k xArms x2) and robot hands used in our demonstration ($3k LEAP Hand V2 x2), the total cost is under $30k. We compare the accuracy and speed of data collection to other commonly used systems: the VR headset and SteamVR. We release videos of our results and instructions to recreate the setup on our website at https://bidex-teleop.github.io

2.1 Related Work

Our work is inspired by the following related work:

Robot Arm Teleop Many common approaches to teleoperation include using joysticks, space mouses [66], vision-based methods, [93, 113], and VR headsets [5, 83, 24] which control arms with inverse kinematics. Joint based teleoperated control has been used in areas such as kinesthetic teaching [10], ABB YuMi [62] and Da Vinci Machines [31], and recently [142, 32] introduced a low-cost version of this with mirroring Trossen Robot arms. GELLO [128] uses light and inexpensive teacher arms that are 3D printed to control full size robot arms, a system we use in our BiDex due to its lost-cost, accurate, portable design.

Robot Hand Teleop The high dimensionality makes tracking human hands particularly difficult. To control robot hands, many vision-based techniques such as [34, 88, 113] do not require specialized equipment but are not that accurate. Shadow Hand developed a professional system that uses SteamVR and two gloves to control two Shadow Hands [101]. Recently, bimanual robot hands and tracking have become accessible for academic labs. Dexcap uses LEAP Hand [105] and tracks the human with gloves and a SLAM-based robot camera. [123] Hato uses a VR headset and controller to control two 6 DOF Psyonic Hands [65, 86]. A key question in controlling robot hands is how to map the human hand configuration to robot hand joints. These papers introduce inverse kinematics based methods that optimize pinch grasps between the human and robot hands [123, 34, 113, 88].

Motion Capture Motion capture and graphics contributions often are useful in the robotic teleoperation domain. Outside-in mocap approaches use external sensing technology to track the human body or other objects in the scene. SteamVR uses external lasers and worn wireless laser receivers. [118] Vicon-based systems use reflective balls and external cameras to track. Inside-out approaches such as XSens [79] or Rokoko suit rely on IMUs on the body but these often drift over time and require recalibration [95, 65]. For hand data, many vision-based approaches such as Frankmocap [97] return MANO [96] parameters which can be converted to robot hand joint angles.

Learning from Expert Demonstrations Recently the robot learning community has seen notable success in learning from demonstrations driven by the development of imitation learning algorithms [72, 18]. Complementing these advances, significant efforts have been made to scale up robotic datasets to facilitate more capable robotic systems [22, 121, 51, 122]. Despite these efforts, acquiring robotics data remains an expensive and challenging endeavor. To address these issues, developments in low-cost hardware have been instrumental in democratizing access to robotic technology, enabling more widespread research and application [114, 142, 32, 19]. However, these systems are primarily focused on simple gripper functionalities; and the challenge of achieving more intricate dexterity and intuitive control in robotic systems motivates our bimanual dexterous teleop system.

2.2 Bimanual Robot Hand and Arm System

We present BiDex , a system that allows any operator to effortlessly teleoperate a bimanual robot hand and arm setup. BiDex is designed to be exceptionally precise, affordable, low-latency, and portable, enabling control of any human-like pair of dexterous hands,

Algorithm 1 Teleoperation Data System

Require: Two robot arms Al, Ar
Require: Two robot multi-fingered hands Hl, Hr
Require: Kinematic models for two arms Kl, Kr
Require: Gloves with fingertip trackers Gl, Gr
Require: Robot hand IK model for fingertips $Q_{l,r}()$
Require: RGB workspace cameras $\{I_c\}$
Require: Number of trajectories to be collected N

Initialize Data buffer $\mathscr{D}$
for trajectory 1:N **do**
    Initialize trajectory $\mathscr{T} = \{\}$
    while task not completed **do**
        Read camera images $\{I_c\}_t$
        Read arm joints Al_t, Ar_t
        Read robot hand joints Hl_t, Hr_t
        Observation $o_t = \{Al_t, Ar_t, Hl_t, Hr_t, \{I_c\}_t\}$
        Read kinematic arm model joints Kl_t, Kr_t
        Read glove fingertip positions Gl_t, Gr_t
        Finger joints $ql_t = Q_l(Gl_t), qr_t = Q_r(Gr_t)$
        Action $a_t = \{Kl_t, Kr_t, ql_t, qr_t\}$
        Add $(o_t, a_t) \mapsto \mathscr{T}$
        Set joints of Al using Kl_t and Ar using Kr_t
        Set joints of Hl using ql_t and Hr using qr_t
    end while
    Add $\mathscr{T} \mapsto \mathscr{D}$
end forreturn Data buffer $\mathscr{D}$

even those with over 20 degrees of freedom. It achieves accurate tracking of the human hand using a Manus VR glove-based system [74] and human arm tracking through a GELLO-inspired system [128]. We outline the process for sending commands and collecting data with bimanual hands in Alg.1. Importantly, our solution functions seamlessly in both tabletop and mobile environments, as it requires no external tracking devices and is highly portable. In Section 2.3, we demonstrate that our system is highly intuitive, precise, and cost-effective compared to widely used methods today, such as VR headsets and SteamVR, across two different pairs of open-source robot hands, LEAP Hand [105] and LEAP Hand V2 [108].

2.2.1 Multi-fingered Hand Tracking

A hand tracking system must deliver accurate, low-latency joint information for the human hand, which has over 20 degrees of freedom. Many current vision-based tracking solutions, such as those using FrankMocap [97, 84] or VR headsets, often face significant inaccuracies due to occlusions and varying lighting conditions, as discussed in Section 2.3. In contrast, recent motion capture gloves that utilize EMF sensors provide significantly more accurate tracking without being overly expensive. They avoid the occlusion issues common in vision-based methods and offer detailed data on the skeletal joint structure of the human hand. Additionally, these gloves can be worn comfortably

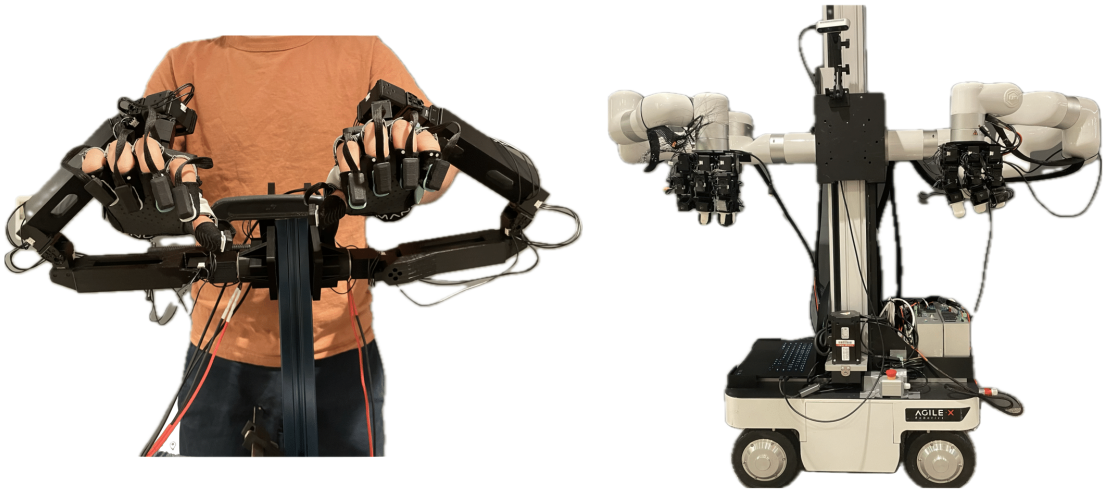


Figure 2.2: **Mobile bimanual teleoperation system** Left: An operator strapped into BiDex . Right: Our bimanual robot setup including two xArm robot arms, two LEAP Hands [105] and three cameras on an AgileX base.

without hindering movement. In BiDex , we opt for the Manus Glove [74], which has demonstrated reliable tracking performance without overheating or suffering from calibration problems as seen with alternatives like the Rokoko gloves [95, 65]. However, mapping this data from the human hand to a robot hand remains challenging due to differences in their morphological structures. How can we translate commands from the operator's fingers to a robot hand that may have a different kinematic configuration?

If the kinematic structure of the robot hand is roughly human-like, one approach would be to directly map the joint angles from human finger joints to those of the robot hand. Although these gloves do not provide true joint angles, they can compute them using an inverse kinematics solver applied to a standard human skeleton. However, if the robot fingers differ significantly in size and proportions from human fingers, the resulting motions may not align properly. The human thumb, in particular, has a complex joint configuration that many robot hands fail to replicate accurately, complicating intuitive thumb control. This can lead to inaccuracies in pinch grasps, negatively impacting the reliability of task performance, as effective manipulation heavily depends on the relative positions of the fingertips.

Previous work [34, 113] has addressed this challenge by ensuring that pinch grasps are consistent between human and robot hands. Wang et. al. [123] demonstrated that effective mapping can be achieved by optimizing the joint positions of the fingertip and penultimate joint (DIP) on each finger in relation to the wrist, ensuring similarity between the human and robot hands through an SDLS IK solver [11]. We employ the Manus gloves [74] using a similar inverse kinematics approach, which allows for precise pinch grasps and proper thumb positioning.

2.2.2 Arm Tracking

Our system must accurately track the human wrist pose to control two robot arms. Traditionally, various methods have been developed by both the motion capture and robotics communities. However, many of these approaches rely on calibrated external tracking devices which either are costly or have high latency. These external tracking devices

are also non-portable, making it hard to scale to mobile systems. Instead, we leverage key insights from ALOHA and GELLO [141, 128] which both use lighter teacher arms attached to the human arm to control a robot arm and hand system. Specifically we follow the GELLO system [128] to teleoperate a full size robot arms. A key question is how to mount this arm-tracking system on a human wrist and hand. If the robot hand is mounted on the arm in a human-like way, then the glove needs to be mounted in a human-like way on the teacher arm to match. However, this orientation means that the human arm will be parallel with the teacher arm and constantly collide with it. This is jarring for the operator and uncomfortable.

In BiDex , we mount the robot hands underneath the arms in the same orientation as if it were a gripper as in Figure 2.2. When mirroring this in the teacher arm, the human arm and teacher arm output are perpendicular to each other and do not collide which is more comfortable. Because of the weight of the motion capture glove, we must adjust the teacher arm to be more robust. This includes adding a strong bearing to the base joint of the teacher arm and adding rubber bands to bias additional joints back to the center of the joint range.

2.2.3 Robot Configurations

We consider two different configurations for the robot arms: tabletop and mobile.

Tabletop Manipulation For our tabletop setup, the robotic arms are positioned to face each other similar to ALOHA [141] while the GELLO teaching arms mirror this configuration. Compared to a side-by-side configuration like in Wang et. al. [123], this setup has three main benefits: 1) the human operators avoid collisions with the teacher arms; 2) the setup allows better visibility of the workspace, which will be otherwise occluded by a side-by-side robot arm configuration; 3) and finally, the robot arms have a wider shared workspace.

Mobile Manipulation BiDex operates without the need for external tracking systems and is lightweight, making it well-suited for mobile settings. The teacher arms are mounted on a compact mobile cart. For the mobile robot, we attach two robot arms to an articulated torso capable of moving upward to reach high objects and downward toward the ground, similar to the PR2 [8], as shown in Figure 2.2. The robotic assembly, which includes the arms and torso, is mounted on an AgileX Ranger Mini, allowing for movement in any SE(2) direction [132, 2]. A secondary operator uses a joystick to control the mobile base and manage task resets.

2.3 Evaluation

We investigate BiDex teleoperating in both the tabletop scenario and in the mobile in-the-wild scenario against a few baselines. For these comparisons, we use LEAP Hand [105] because it is an open source easy to assemble robot hand that is readily attainable by any robotics lab. We also use BiDex with the more recent LEAP Hand v2, which is made of a combination of rigid and soft material and capable of performing more complex tasks.

2.3.1 Baseline Teleoperation Approaches

We compare BiDex with the following alternative teleoperation systems:

Figure 2.3: **All Tasks**: Teleoperation of the mobile robot systems with BiDex . **Top**: Picking up trash from a table and discarding it into a bin. **Bottom**: Grasping a chair and moving it to align with a table.

Vision based VR Headset In recent years, the accessibility of low-cost VR headsets using multi-camera hand tracking has made them popular for teleoperation such as in [24, 83] As a baseline we use the Apple Vision Pro which returns both finger data similar to MANO parameters [84] and wrist coordinate frame data. The finger data is used in the same way as with BiDex through inverse kinematics-based retargeting and commanded onto the robot hands. The wrist data is reoriented, passed through inverse kinematics and the final joint configuration is commanded to the arms.

SteamVR Tracking SteamVR, commonly used in the video gaming community has also seen recent interest in the robotics community from industry [101] and academia alike. [73, 1] It uses active powered laser lighthouses that must be carefully placed around the perimeter of the workspace. Wearable pucks with IMUs and laser receivers are worn on the body of the operator. In our experiment the operator wears one tracker on each wrist and one tracker on their belly. The wrist position is determined relative to the belly pose, mapped to the robot arm, and the joint angles are computed using inverse kinematics. The hand tracking gloves are the same as in BiDex .

2.3.2 Choice of Dexterous End-Effectors

We consider two different dexterous end-effectors for our teleoperation system:

Leap Hand LEAP Hand [105] is a low-cost, easy-to-assemble robot hand with 16 DOF and 4 fingers. LEAP Hand introduces a novel joint configuration that optimizes for dexterity as well as human-like grasping. We use this hand for many experiments in the paper as it is a readily available dexterous hand available for comparison studies.

LEAP Hand V2 We would like a hand that is smaller and more compliant than LEAP Hand. LEAP Hand V2 is crafted to mimic the suppleness and strength of the human hand with fingers that have a 3D-printed flexible outer skin paired with a sturdy inner framework resembling bones. These fingers do not break but instead bend and

	Completion Rate			Time Taken		
	Handover	Cup Stacking	Bottle Pouring	Handover	Cup Stacking	Bottle Pouring
`Vision Pro VR`	60	40	70	21.6	38.8	35.5
`SteamVR`	80	85	60	17.5	16.5	15.5
`BiDex`	95	75	85	6.5	15.5	14.9

Table 2.1: **Tabletop Teleoperation**: We compare BiDex on the handover, cup stacking, and bottle pouring tasks to two baseline methods, SteamVR and Vision Pro. BiDex enables more reliable and faster data collection, especially for harder tasks like bottle pouring.

	Completion Rate			Time Taken		
	Chair Pushing	Box Carry	Clear Trash	Chair Pushing	Box Carry	Clear Trash
`Vision Pro VR`	75	75	50	15.0	33.7	79.8
`BiDex`	95	95	75	16.4	29.7	74.6

Table 2.2: **Mobile Teleoperation**: Completion rate and time taken averaged across 20 trials using a mobile bimanual system with LEAP Hand [105], for different tasks. BiDex is versatile and compact enough to be adopted to successfully collect data for mobile tasks.

flex upon impact. We also introduce an active articulated palm which integrates two motorized joints, one spanning the fingers and another for the thumb, enabling natural tight grasping. LEAP Hand V2 contains 21 degrees of freedom and is sized to resemble a human hand, easy to assemble and is economical. Because of the human-like size and kinematics, it is easy to retarget to and can complete many more dexterous tasks successfully.

2.3.3 Task Descriptions

For the tabletop and the mobile experiments, we consider the following tasks:

For the **Tabletop** experiments, we consider the following tasks: In *Handover*, the robot picks up a pringles can and passes it from its right hand to its left hand in the air. In *Pour*, the robot pours from a glass bottle in one hand into a plastic cup held by the other hand. In *Tabletop Cup Stack* the agent stacks one cup into another cup in the other hand.

For the **Mobile** experiments, we consider the following tasks: In *Transport Box*, the agent moves a box from one table to another table using two hands. In *Mobile Chair Push*, the agent needs to grasp a chair, and then align it with a table. In *Clear Trash*, the robot clears trash from the table into a dustbin. We visualize the chair push and clear trash tasks in Figure 2.3.

2.3.4 Bimanual Dexterous Teleoperation Results

For the teleoperation experiments, we have the following findings:

BiDex provides more stable arm tracking. The teacher arm system makes BiDex highly reliable, with minimal jitter, low latency and high uptime, making it ideal for arm tracking. The teacher arm is lightweight and doesn't impede the user more than the gloves' weight. The kinematic feedback from arm resistance is subtle but helps operators navigate around arm singularities intuitively. As shown in Tables 2.1 and 2.2, BiDex achieves a higher completion rate in less time for teleoperators.

	Can Handover	Cup Stacking	Bottle Pouring
Leap Hand	7/10	14/20	16/20

Table 2.3: **Imitation learning**: We train ACT from [141] using data collected by BiDex and find that our system can perform well even in this 44 dimension action space. This demonstrates that our robot data is high quality for training robot policies.

In contrast, the Vision Pro often experiences jittery arm tracking, complicating the teleoperation of more demanding tasks. While low-pass filtering can mitigate this issue somewhat, it introduces undesirable latency. Occasionally, the system may stop functioning entirely, which can be disconcerting for users.

The SteamVR system offers wireless connectivity, allowing users to be untethered, and it generally provides accurate tracking. However, it can experience brief episodes of high latency or disconnections every 5-10 minutes, which can be jarring. Notably, the SteamVR system cannot be used in mobile settings due to the need for external tracking lighthouses to be set up around the teleoperation environment.

BiDex provides more accurate hand tracking. With BiDex , fingertip tracking is highly accurate when using the Manus glove. When mapping to different robots, only minor adjustments to inverse kinematics are required for operators with varying hand sizes. The accuracy of abduction and adduction at the MCP joint remains dependable across different conditions. These benefits are particularly crucial in LEAP Hand V2 , where precision is essential for executing more complex tasks.

In contrast, the Vision Pro can struggle with hand size variability under different lighting conditions, making the retargeting process to robot hands more challenging. Additionally, finger abduction and adduction estimates can be impacted by occlusions, which complicates the performance of intricate tasks. The latency is noticeable and can pose a significant issue for teleoperation.

2.3.5 Training Dexterous Visuomotor Policies with BiDex

To verify that the data that is collected by our system is high quality and useful for machine learning we train single task closed loop behavior cloning policies.

Specifically, we train an action chunking transformer from [141] with a horizon length of 16 at 30hz using pretrained weights from [23] on around 50 demonstrations. The state space is the current joint angles of the robot hand and the images from the camera. The action space in the case of LEAP Hand is 16 dimensions for each hand and 6 dimensions for each arm for a total of 44 dimensions. During rollouts, the behavior of the policies are very smooth, exhibiting the high quality of the teleop data. In tasks such as the YCB Pringles can [12] handover, we even see good generalization of the policy to different initial locations of the can.

2.3.6 Extreme Dexterity using LEAP Hand V2

To push BiDex to its dexterous limits, we use LEAP Hand V2 : an extremely dexterous 21 DOF hybrid low-cost hand which is explained in Section 2.3.2.

To do this, we show a variety of very challenging tasks as in Figure 2.1 and Figure 2.4. These tasks include pouring, scooping, hammering, chopstick picking, hanger picking,

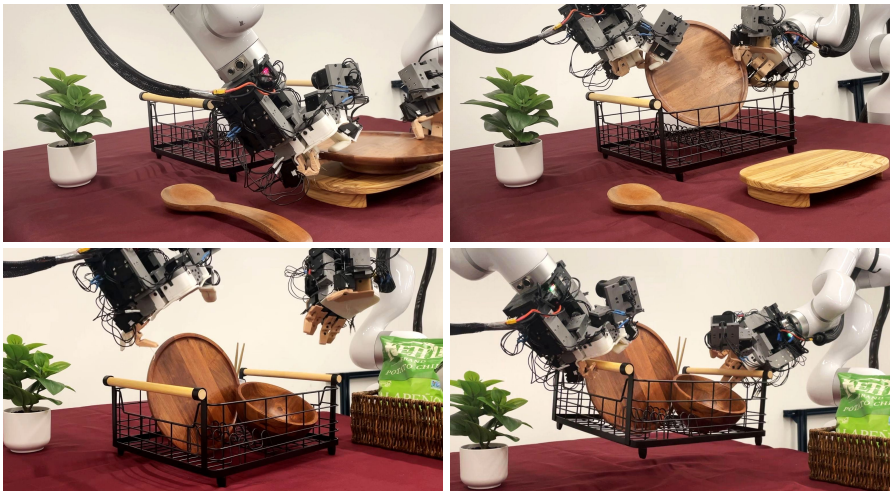


Figure 2.4: **Clearing the Dishes**: In this task, we use BiDex to perform a long horizon task to place bowls and spoons into a drying rack and lift the drying rack away from the table.

	Drill	Lift Pot	Bottle Pouring	Plate Pickup
Leap Hand v2	15/20	15/20	15/20	13/20

Table 2.4: **Imitation learning LEAP v2**: We also train ACT using LEAP Hand v2 and show task completion on more dexterous tasks.

picking up basket, drilling, plate pickup and pot picking. In these experiments we find that BiDex scales well to this high DOF hand and it feels very natural to control this soft-rigid robot hand. We provide Table 2.4 and video results on our website at `https://bidex-teleop.github.io`

2.4 Discussion

In this paper, we introduce BiDex , a portable, low-cost and extremely accurate method for teleoperating a bimanual, human-like robot hand and arm system. We demonstrate the system's applicability to both a tabletop and a mobile setting and show its efficiency in performing bimanual dexterous tasks in comparison to alternative approaches including SteamVR and Vision Pro. Nevertheless, our BiDex is not without limitations. Due to the lack of haptic feedback, the human operator has to rely on visual feedback for teleoperation and cannot feel what the robot hand is feeling. Additionally, they cannot exert intricate force control and can only control the kinematics of the robot hand and arm which can make it challenging for fine-grained manipulation tasks. A promising direction in the future would be to integrate haptic feedback into our system which will unlock further potential for collecting extreme dexterity data.

Chapter 3

FACTR: Teleoperation and Acting with Force Feedback

Contact-rich tasks are an integral part of daily life, from lifting a box and rolling dough to cracking an egg or opening a door. These tasks, while seemingly simple, involve a complex interplay of forces and require precise adjustments based on force feedback. Humans rely heavily on this force feedback to generalize across tasks and objects, adapting seamlessly to variations in visual appearances and geometries. However, in robot learning, force information remains underutilized, even though it is readily available on many modern robotic arms, such as the Franka Panda and the KUKA LBR iiwa. Instead, most data-driven methods, including those using Behavior Cloning (BC), focus primarily on visual feedback for both data collection and policy learning, overlooking the critical role of force. This limited use of force information hinders the vision-only policies' ability to generalize to novel objects. For instance, in tasks like lifting up a box with two arms, the primary factor influencing the action is the object's geometry, while attributes such as color or texture are irrelevant. In such cases, force feedback provides a clear signal for mode switching, such as detecting when contact is established, which can facilitate object generalization compared to relying solely on vision.

One of the main reasons for the under-utilization of force feedback in robot learning is the lack of an intuitive and *low-cost* teleoperation system that can capture force feedback during data collection itself. Recently, low-cost leader-follower systems have become popular for teleoperation, offering intuitive control of robot arms by mirroring the joint movements of the leader arm controlled by a teleoperator to the follower arm [141, 129]. However, these systems are typically passive (leader arm joints are not actuated) and unilateral (the leader arm does not receive information from the follower arm). This makes teleoperation difficult for dynamic, contact-rich tasks where precise force adjustments are necessary [82]. To overcome this limitation, we present a bilateral low-cost teleoperation system that provides force feedback by actuating motors in the leader arm joints based on external joint torques transmitted from the follower arm (Fig. 3.1 Left). By actuating the motors, we also provide active gravity compensation and resolve the kinematic redundancy due to the redundant degrees of freedom of the arm. These enhancements improve the teleoperation experience, leading to a 64.7% increase in the task completion rate, a 37.4% reduction in completion time, and an 83.3% improvement in subjective ease of use across four evaluated contact-rich tasks.

The second challenge lies in effectively incorporating robot force information into

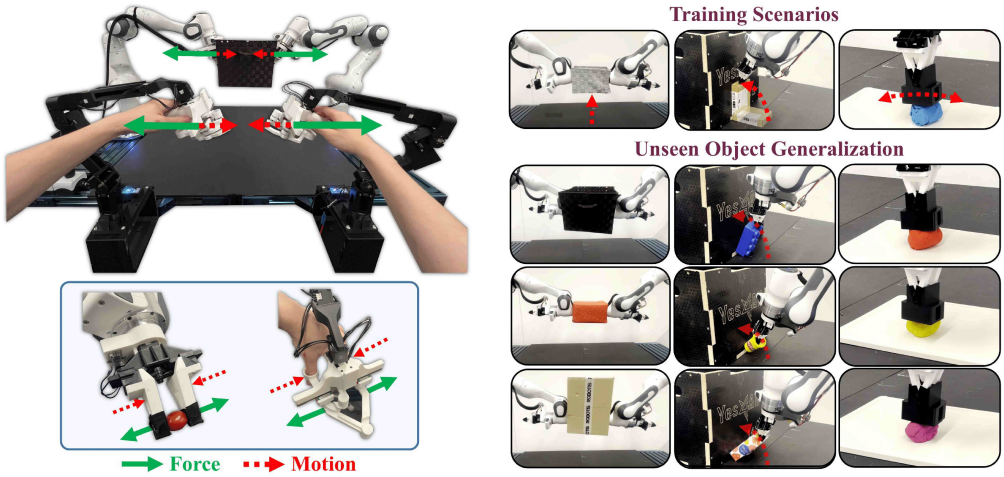


Figure 3.1: **FACTR.** We present *Force-Attending Curriculum Training* (FACTR) – a system that leverages robot external joint torques for both teleoperation and improving policy generalization for complex contact-rich tasks. [Left] Our low-cost leader-follower setup employs actuated servo motors to enable force feedback in both the leader arm and gripper, improving teleoperation success rate, completion time, and ease of use. [Right] FACTR's behavior cloning policy utilizes robot force information to enhance performance and generalization across objects with diverse geometries and textures in contact-rich tasks. Video results, codebases, and instructions at `https://jasonjzliu.com/factr/`

policy learning. Although recent methods such as diffusion policy [18] and action chunking transformers [141] achieve impressive results for fine-grained manipulation, they often fail to generalize to unseen objects with variations in object visual appearances and geometries. Humans, on the other hand, can disregard irrelevant visual details once contact is established and rely solely on force feedback to perform tasks such as lifting a box or rolling dough. Therefore, to improve generalization, we seek to incorporate force input into autonomous robot policies. However, making effective use of force information in policy learning is challenging, as policies often overfit to using visual modality [124], effectively disregarding force data. This issue arises because contact force signals are typically less discriminative, often remaining near zero for extended periods when the arm is not in contact with the environment during an episode. Hence, without proper care during training, policies tend to ignore force input and rely primarily on visual information. We empirically analyze this effect in Sec. 3.4.3 and Fig. 3.9.

To mitigate this issue, we propose Force-Attending Curriculum Training (FACTR), a curriculum training strategy designed to improve the policy's ability to effectively leverage force information. FACTR systematically reduces the reliance on visual information during training by applying operators such as Gaussian blurring or downsampling with varying scales to visual inputs. A scheduler gradually decreases the blurring scale and increases the fidelity of the visual inputs. Intuitively, this approach encourages the policy to focus more on force input during initial training phases and gradually balances force with visual inputs as training progresses. We ground this intuition with a theoretical analysis on a simplified scenario through the framework of Neural Tangent Kernels [43]. We explore FACTR in both the pixel space and latent space, testing various operators and scheduling strategies. Our experiments show that FACTR improves the success rate for unseen objects by an average of 40.0% in four challenging contact-rich tasks

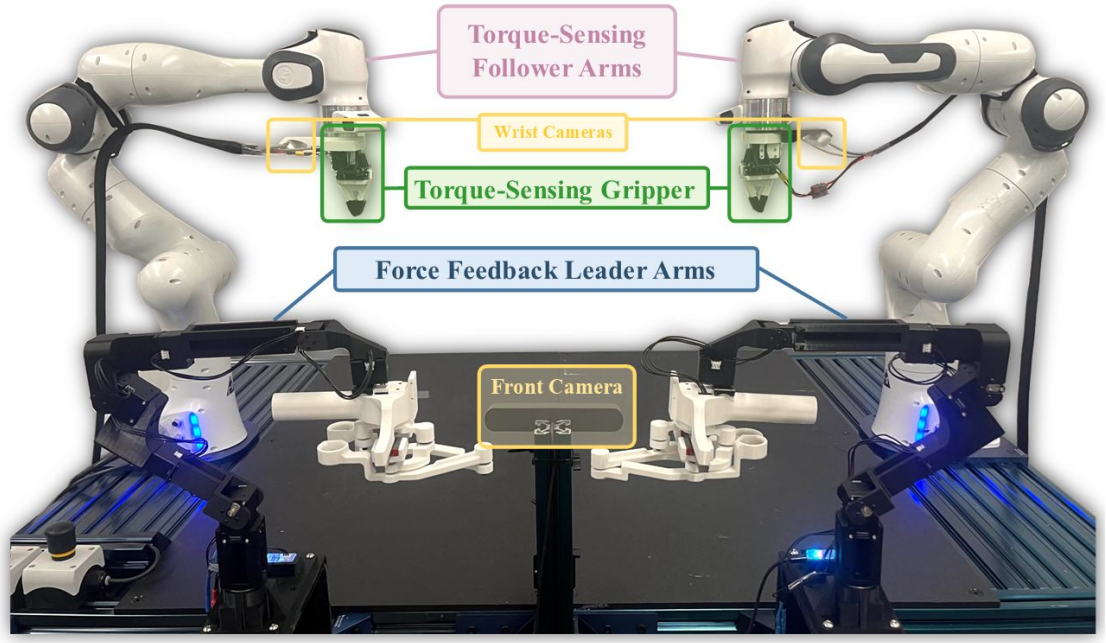


Figure 3.2: **Our low-cost bimanual teleoperation system with force-feedback.** The system features two actuated leader arms, two follower arms with external joint torque sensors (such as the Franka Panda and the KUKA LBR iiwa), a front camera and two wrist cameras.

(Fig. 3.1 Right) – box lifting, prehensile pivoting, fruit pick-and-place, and dough rolling – showcasing the efficacy of our force-attending curriculum training.

In summary, our contributions are as follows.

- **Low-Cost Teleoperation with Force Feedback:** We design a low-cost bilateral leader-follower teleoperation system with force feedback, gravity compensation, and redundancy resolution, demonstrating a 64.7% improvement in task completion rate and an 83.3% enhancement in ease of use for teleoperation through a user study.
- **Force-Attending Curriculum Training:** We propose FACTR, a curriculum training framework that better learns to use force feedback in policy learning and achieves better generalization capability to object visual appearances and geometries. Evaluated on four challenging contact-rich tasks, FACTR improves the performance of autonomous policies by 40.0% compared to policies with force information as part of input but trained without FACTR.

3.1 Related Work

Our work is closely related to the following related work:

Imitation Learning with Force Imitation learning has recently experienced significant advancements, driven by the development of more effective algorithms that leverage demonstrations to train robotic policies [72, 18]. Although traditional approaches primarily rely on visual and joint position inputs, many real-world tasks require explicit force feedback to improve stability, adaptability, and safety [75, 37]. Recent work has applied learning methods to train with demonstrations that incorporate gripper force or tactile

signals, resulting in policies capable of handling small, fragile objects and performing contact-intensive tasks such as vegetable peeling [68, 131, 61]. However, utilizing force data from the robot arms, such as joint torques, remains underexplored. One approach involves using an end-effector force sensor to estimate compliance parameters or virtual position targets through kinesthetic teaching and force tensors [38, 17]. Another method infers a 6D wrench for low-level control by integrating torque sensing into a diffusion policy [130].

However, naively incorporating force feedback into policy learning can lead to overfitting to visual information, causing the policy to disregard force input. FoAR [36] explicitly predicts contact and non-contact phases to regulate the fusion of vision and force modalities, which requires additional data labeling. We propose FACTR to effectively incorporate force and vision input into policy through a curriculum, enabling policies to leverage force for improved object generalization.

Low-Cost Teleoperation Systems with Force Feedback Parallel to advances in imitation learning, significant efforts have been made to collect low-cost and high-quality data with hand-held grippers [114, 20] or leader-follower systems [141, 129, 107]. Hand-held grippers naturally provide force feedback to the operator, but they do not directly record force data. Recent work has added force sensors to hand-held grippers to address this limitation [68]. However, hand-held grippers are in general limited by the kinematic differences between humans and robots, resulting in commands that might be unachievable for the robots. Although the leader-follower systems are not prone to this limitation, they often lack force feedback, impairing their effectiveness in contact-rich tasks. Recently, Kobayashi et al. [53] implemented a bilateral leader-follower teleoperation system where in addition to the follower following the joint positions of the leader, the leader also gets an additional torque if there is a difference in its joint position from that of the follower. However, when the follower arm is in motion without contact, this system causes the operator to experience inertial, frictional, and other dynamic forces of the follower, reducing the ease of use and precision of the system [111]. Our approach introduces an alternative bilateral teleoperation method by relaying only external joint torques from the follower arm back to the leader arm, providing force feedback without impairing operational precision.

3.2 FACTR Low-Cost Bilateral Teleoperation

Leader-follower systems, such as GELLO or ALOHA [129, 141], offer a simple and cost-effective solution to teleoperation in manipulation tasks. These systems feature kinematically equivalent leader and follower arms, allowing intuitive control through joint space mapping, where the leader's joint positions are mirrored as targets for the follower. This setup lets users naturally feel the follower arms' kinematic constraints. However, most implementations lack force feedback, preventing users from sensing the geometric constraints of the environment, which is crucial for teleoperating contact-rich tasks [82]. Instead, those leader arms are mostly passive, lacking active motor torque actuation, despite being equipped with servo motors capable of actuation. Furthermore, the lack of active torque means the leader arms require external structural frames and rubber bands or strings to achieve gravity compensation, reducing portability [141].

In this paper, we aim to fully leverage the servo motors in the leader arm and gripper to achieve force-feedback enabled teleoperation with affordable hardware. Similarly

to GELLO [129], our leader arms use off-the-shelf servos and 3D-printed components, forming a scaled-down but kinematically equivalent version of the follower arms, as shown in Fig. 3.2. By actuating the servo motors, we introduce force feedback, customizable redundancy resolution through nullspace projection, gravity and friction compensation, and joint limit avoidance. These functionalities augment the teleoperation experience while still using low-cost hardware to provide functions that are usually only available with much more expensive teleoperation devices. Please see Appendix B.2 for a detailed Bill of Materials.

3.2.1 Force Feedback

Force feedback provides the operator with a tangible sense of interaction with the environment, allowing more intuitive and delicate manipulation, especially in contact-rich tasks or tasks with limited visual feedback [82]. We implement a control law that relays external joint torques sensed by the follower arm to the leader arm, allowing the operator to feel the physical constraints experienced by the follower arm:

$$\tau_{feedback} = \mu_f \mathbf{K}_{f,p} \tau_{ext} - \mathbf{K}_{f,d} \dot{\mathbf{q}} \tag{3.1}$$

where μ_f is a scalar constant, τ_{ext} is the external joint torque sensed by the follower arm, $\mathbf{K}_{f,p}$ and $\mathbf{K}_{f,d}$ are the PD gains for the force feedback, respectively. Here, $\mathbf{K}_{f,p}$ is calculated as the ratio between the maximum torque of the leader and that of the follower, and $\mathbf{K}_{f,d}\dot{\mathbf{q}}$ helps reduce oscillations in the leader arm when the follower arm is in contact. We note that τ_{ext} is a readily available measurement in various collaborative robot manipulators, such as the Franka Panda and the KUKA LBR iiwa. In particular, we implement mediated force feedback by scaling down τ_{ext} with μ_f, which has been shown to improve the accuracy of the operation while reducing the cognitive load of the operator [144]. Furthermore, we highlight that our implementation only transmits external forces from the follower to the leader; as a result, the operator does not experience the internal friction and inertia of the follower arm during motion, providing a clearer perception of the environment [111].

In addition, we implement force feedback for the parallel-jaw gripper. Since our servo-based gripper does not contain an external force sensor, we utilize the present current reading of the gripper servo to provide force feedback as follows:

$$\tau_{h,t} = \alpha(-k_h I_{g,t}) + (1-\alpha)\tau_{h,t-1} \tag{3.2}$$

where $\tau_{h,t}$ is the force feedback torque sent to the gripper leader device, $I_{g,t}$ is the present current reading from the follower gripper, and α is the smoothing factor for the EMA filter. Our system sets $\alpha = 0.1$ which provides a good user experience.

3.2.2 Customizable Redundancy Resolution

For kinematic redundant manipulators, without regulating the joint space, the manipulator tends to drift into undesirable configurations under the influence of gravity during teleoperation. Approaches like Gello [129] rely on mechanical components, such as springs, to regularize the joint space. However, these components introduce non-uniform, configuration-dependent wrenches at the end-effector, resulting in an unintuitive teleoperation experience. In addition, using mechanical joint regularization effectively prevents

the user from setting custom joint regularization targets for redundancy resolution. In confined-space manipulation settings, the inability to control the joint regularization target can impair the arm's reachability, as demonstrated in Fig. 3.3.

In contrast, our proposed method leverages the following null-space projection control law to regulate joint positions [50], which stabilizes the joint-space at any user-defined desirable posture without imposing additional end-effector wrenches, regardless of the arm's configuration:

$$\tau_{null} = \left(\mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J}\right)\left(-\mathbf{K}_{n,p}\left(\mathbf{q} - \mathbf{q}_{rest}\right) - \mathbf{K}_{n,d}\dot{\mathbf{q}}\right) \tag{3.3}$$

where $\mathbf{J}$ is the manipulator Jacobian matrix, $\mathbf{q}_{rest}$ is a user-defined resting posture configuration, $\mathbf{K}_{n,p}$ and $\mathbf{K}_{n,d}$ are the PD gains for the null space projection. Note that $\left(\mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J}\right)$ is the null-space projector.

3.2.3 Gravity Compensation

To ensure the leader arms remain stationary, allowing the user to easily pause teleoperation, we implement gravity compensation. This is achieved by modeling the dynamics of the leader arm and computing the joint torques required to counteract dynamic forces using the recursive Newton-Euler algorithm (RNEA) for real-time inverse dynamics [70].

$$\tau_{grav} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \mathrm{RNEA}(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}}) \tag{3.4}$$

where $\mathbf{M}(\mathbf{q})$ is the mass (or inertia) matrix, $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$ is the Coriolis and centrifugal matrix, and $\mathbf{g}(\mathbf{q})$ is the gravity vector.

3.2.4 Additional Compensation and Controls

To reduce the perceived friction in the leader arm during teleoperation, our system provides friction compensation $\tau_{friction}$. Furthermore, since the leader arm joints lack mechanical joint limits, we implement an artificial potential based control law to prevent users from exceeding joint limits of the follower arm in order to respect the workspace of the follower arm. Finally, for bi-manual follower arms, the system uses Riemannian Motion Policies [92] for dynamic obstacle avoidance between the two follower arms. Please refer to Appendix B.3 for more details.

3.2.5 Overall Control Law for the Leader Arm

In summary, the control torques are defined as follows:

- $\tau_{feedback}$ relays external forces from the follower arm back to the leader arm, allowing the operator to sense the geometric constraints of the environment.
- τ_{null} resolves kinematic redundancy by regulating the joints at a user-defined rest posture in the null-space.
- τ_{grav} provides gravity compensation for the leader arm.
- $\tau_{friction}$ compensates for the leader arm joint frictions to enable smoother teleoperation.

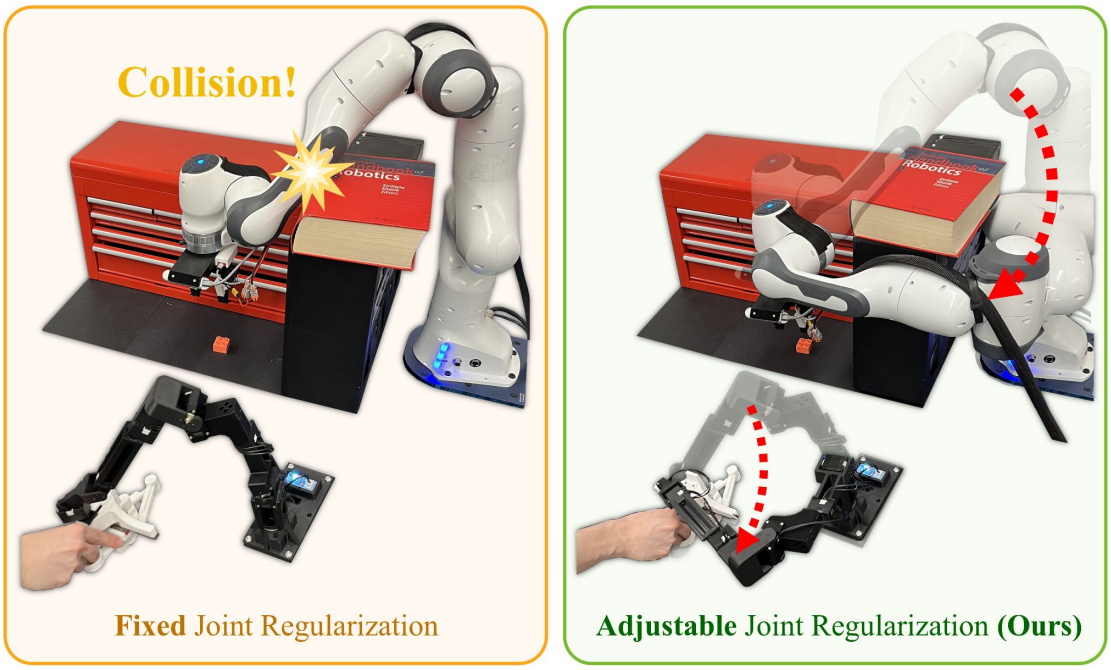


Figure 3.3: **Customizable Joint Regularization** [Left] Without the flexibility to define the resting joint configuration q_{rest}, the arm's reachability is restricted, leading to collisions in confined spaces. [Right] Our leader arm allows the user to define custom resting joint q_{rest}, helping the follower arm reach targets in confined spaces.

- τ_{limit} prevents the joints of the leader arm from violating the joint position limits of the follower arm.

The resulting combined torque applied to the servo motors of the leader arm is defined as follows:

$$\tau = \tau_{feedback} + \tau_{null} + \tau_{grav} + \tau_{friction} + \tau_{limit} \tag{3.5}$$

3.3 FACTR: Force-Attending Curriculum Training

Naively incorporating robot force data into policy learning does not necessarily ensure policy improvement. Contact force signals often provide limited discriminative information for the policy, as it remains close to zero for significant periods when the arm is not interacting with the environment during an episode. As a result, the policy tends to disregard force input and rely predominantly on visual information, as empirically analyzed in Sec. 3.4.3 and Fig. 3.9.

To fully leverage the robot force data collected from our teleoperation system, we introduce Force-Attending Curriculum Training (FACTR), a training strategy designed to effectively integrate force information into policy learning. FACTR applies operators like Gaussian blur or downsampling to corrupt visual information, where the amount of visual corruption decreases throughout training. The curriculum intuitively encourages contribution from the force modality at the start of training. We ground this intuition with a theoretical analysis on a simplified scenario through the framework of Neural Tangent Kernels [43].

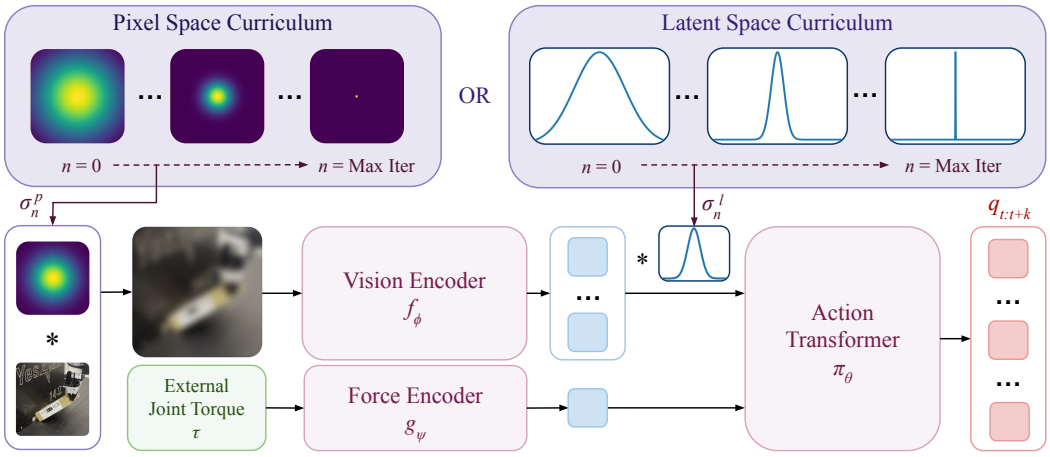


Figure 3.4: **FACTR** allows our policy to better integrate force information without overfitting to visual information, resulting in better generalization to objects with unseen visual appearances and geometries. Our policy takes as inputs RGB images I and external joint torque τ, which are then tokenized by a vision encoder and a force encoder before fed into an action transformer to regress joint position targets $q_{t:t+k}$. FACTR applies a blurring operator of scale σ_n in either pixel or latent space, initialized at a large value then gradually decreased through the training.

In this section, we first present the base policy model used for learning from teleoperated demonstrations, and then motivate and describe FACTR, our curriculum training approach. Our overall method is summarized in Algorithm 2 and Fig. 3.4.

3.3.1 Problem Statement and Base Model

We consider a policy $\pi_\theta(\cdot \mid \cdot)$ that produces a chunk of future actions of length k $\hat{q}_{t:t+k}$ (joint positions) given (i) a visual observation I_t (image at time t), and (ii) an external joint torque reading τ_t. Our goal is to learn π_θ via behavior cloning (BC) from a dataset of expert trajectories $\mathscr{D}$. Each trajectory in $\mathscr{D}$ comprises tuples $\big(I_t, \tau_t, q_t\big)$, where q_t is the ground-truth (expert) joint position target at time t. We let $\hat{q}_{t:t+k}$ be the *predicted* future joint position targets over the next k time steps. The loss is defined by:

$$\mathscr{L} \;=\; \mathrm{MSE}\big(\hat{q}_{t:t+k}, q_{t:t+k}\big), \tag{3.6}$$

where $q_{t:t+k}$ are the expert's future joint position targets and $\hat{q}_{t:t+k}$ are the policy's predictions.

Our policy π_θ is based on an encoder-decoder transformer that integrates vision and force modalities. Visual observations and force readings are converted into tokens, fed to the *encoder*, then decoded into action tokens through cross attention.

A pre-trained vision transformer (ViT) [25, 23] is used to encode an input image I_t into a sequence of *vision tokens* $\mathbf{z}_t^V \in \mathbb{R}^{M_v \times d}$ for some number of tokens M_v and embedding dimension d. An MLP-based force encoder is applied to the joint torque τ_t, resulting in a single *force token*: $\mathbf{z}_t^F \in \mathbb{R}^{1\times d}$. The tokens are concatenated to form the model input:

$$\mathbf{X}_t \;=\; \big[\mathbf{z}_t^V; \mathbf{z}_t^F\big] \;\in\; \mathbb{R}^{(M_v+1)\times d}.$$

Then, a transformer encoder Enc processes $\mathbf{X}_t$ via multiple self-attention and feed-forward layers:

$$\mathbf{H}_t^E \;=\; \mathrm{Enc}\big(\mathbf{X}_t\big) \;\in\; \mathbb{R}^{(M_v+1)\times d}.$$

This yields the *encoded* vision and force tokens.

For the decoder, we introduce *k action tokens*, $\mathbf{A} \in \mathbb{R}^{k \times d}$. A transformer decoder Dec refines these tokens through self-attention and cross-attention to $\mathbf{H}_t^E$:

$$\mathbf{H}_t^D \;=\; \mathrm{Dec}\big(\mathbf{A},\, \mathbf{H}_t^E\big).$$

During cross attention, each action token attends to both vision and force representations. If we split $\mathbf{H}_t^E$ into its vision (V) and force (F) parts, the cross-attention weights for each layer l can be decomposed as follows. For simplicity of notation, assume these weights are already averaged over multiple heads:

For the vision part:

$$\alpha_V^{(l)} \;=\; \mathrm{softmax}\left((\mathbf{A}^{(l)}\mathbf{W}^{Q(l)})\, (\mathbf{H}_{t,V}^{E(l)}\mathbf{W}^{K(l)})^\top / \sqrt{d} \right),$$

For the force part:

$$\alpha_F^{(l)} \;=\; \mathrm{softmax}\left((\mathbf{A}^{(l)}\mathbf{W}^{Q(l)})\, (\mathbf{H}_{t,F}^{E(l)}\mathbf{W}^{K(l)})^\top / \sqrt{d} \right).$$

These $\alpha_V^{(l)}$ and $\alpha_F^{(l)}$ measure how strongly each action token attends to vision vs. force tokens at layer l, and will be the main source of analysis in Sec. 3.4.3.

Finally, we project the decoder output $\mathbf{H}_t^D$ to action space, which represents joint position targets for the follower arm:

$$\hat{q}_{t:t+k} = \mathrm{MLP}\big(\mathbf{H}_t^D\big) \;\in\; \mathbb{R}^{l \times d_a}.$$

where d_a is the dimension of the action space. Substituting $\hat{q}_{t:t+k}$ into Eq. 3.6 gives the full BC objective. Please see Appendix B.4 for the detailed policy architecture and training hyperparameters.

3.3.2 Force-Attending Curriculum

Through experiments, as shown in Sec. 3.4.3 and Fig. 3.9, we found that naively concatenating force data to the policy observation during training often results in policies that neglect force input, failing to leverage force input to the fullest extent. To address this, we employ a *curriculum* that gradually unveils detailed visual information, encouraging the model to learn to utilize force first. Specifically, we define two operators: $\beta_P(I, \sigma_n)$ for the pixel space, and $\beta_L(z, \sigma_n)$ for the latent space, where σ_n is a scale parameter (e.g. the standard deviation of a Gaussian kernel or the kernel size of a max pooling operator) that is updated over the course of training for N total gradient steps. During training, we apply the pixel-space operator β_P to image I_t or β_L to visual latent tokens z_t^V.

Intuitively, the operators make visual inputs or latent tokens close in the metric space, thus encouraging more contribution from the force modality, particularly at the start of the training. Consider the limit $\sigma \to \infty$, each visual input converges to approximately the same tensor. Hence, the model can only learn a single global output for all visual inputs. Thus, at the early stage of the curriculum, the gradient updates focus more on using the force information and updating the force encoder to maximally differentiate between inputs.

Algorithm 2 Force-Attending Curriculum Training (FACTR)

Given: Expert dataset $\mathscr{D}$; action chunking size k; total training steps N; Pixel-space operator $\beta_P(I, \sigma)$; latent-space operator $\beta_L(z, \sigma)$; Scheduler defining σ_n for $n = 1 \dots N$
Initialize pre-trained ViT f_ϕ, force MLP encoder g_ψ, and action-chunking transformer π_θ
for iteration $n = 1 \dots N$ **do**
    Sample $\left(I_t, \tau_t, q_t\right)$ from $\mathscr{D}$
    $\sigma_n \leftarrow \text{Scheduler}(n)$ // get current scale
    if pixel-space curriculum **then**
        $I_t \leftarrow \beta_P\left(I_t, \sigma_n\right)$
    end if
    $\mathbf{z}_t^V \leftarrow f_\phi(I_t)$ // vision tokens
    if latent-space curriculum **then**
        $\mathbf{z}_t^V \leftarrow \beta_L\left(\mathbf{z}_t^V, \sigma_n\right)$
    end if
    $\mathbf{z}_t^F \leftarrow g_\psi(\tau_t)$ // force token
    $\hat{q}_{t:t+k} \leftarrow \pi_\theta\left(\mathbf{z}_t^V, \mathbf{z}_t^F\right)$
    $\mathscr{L} = \text{MSE}\left(\hat{q}_{t:t+k}, q_{t:t+k}\right)$
    Update ϕ, ψ, θ using ADAM
end for

3.3.3 Curriculum Operators

We consider two types of operators: Gaussian blur and downsampling.

For the Gaussian blur, we define the 2D kernel G_σ as:

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

The operator $\beta_P(I,\sigma)$ applies this kernel using the 2D convolution operator $*$:

$$\beta_P(I,\sigma) = I * G_\sigma$$

For the 1D Gaussian blur, the kernel g_σ is defined as:

$$g_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

The operator $\beta_L(z^V,\sigma)$ similarily applies this kernel using 1D convolution:

$$\beta_L(z^V,\sigma) = z^V * g_\sigma$$

For downsampling, we use MaxPool followed by nearest interpolation. In 2D, the pixel-space operator $\beta_P(I)$ is:

$$\beta_P(I) = \text{NearestInterp}(\text{MaxPool2D}(I))$$

In 1D, the latent-space operator $\beta_I(z^V)$ is the same except that a MaxPool1D is used.

By gradually reducing σ_n, the curriculum ensures that the model focuses first on force tokens, and then incorporates visual information in the later stage of the training. This produces a policy π_θ that more robustly fuses force and vision for control, alleviating the issue of overfitting to the vision modality.

Theoretical Analysis: We analyze the effects of Gaussian blur as an example of a curriculum operator through the framework of Neural Tangent Kernels (NTK) [43]. Although we consider a simple two-layer model here, the intuition applies to more complex architectures like vision transformers (ViTs), which have a more sophisticated NTK [9]. The formal theoretical analysis is presented in Appendix B.1.

3.3.4 Curriculum Schedulers

Over the course of training (indexed by $n = 1, \ldots, N$), we adjust σ_n via a scheduler to control the information released from the visual branch. Given a initial scale σ_0, we consider the following schedulers:

Decay Type	**Scheduler Equation**
Constant	$\sigma_n = \sigma_0$
Linear	$\sigma_n = \sigma_0 \left(1 - \frac{n}{N}\right)$
Cosine	$\sigma_n = \frac{\sigma_0}{2}\left(1 + \cos\left(\frac{n\pi}{N}\right)\right)$
Exponential	$\sigma_n = \sigma_0 \cdot \alpha^n, \quad \alpha > 1$
Step	$\sigma_n = \sigma_0\left(1 - \frac{1}{d_{\text{steps}}}\left\lfloor \frac{n}{N/d_{\text{steps}}} \right\rfloor\right), \quad d_{\text{steps}} > 1$

Furthermore, we warm-up the curriculum by fixing the scale to σ_0 for certain gradient steps, and adjust the decay formula to account for this duration. The rationale behind this step is to warm-up the randomly initialized force encoder with relatively low visual information.

3.4 Evaluation

3.4.1 Experimental Setup

We setup four contact-rich tasks, which are illustrated in Fig. 3.5 along with the training and testing objects of various shapes and visual appearances. For all tasks, we use Franka Panda arm(s) with OpenManipulator-X gripper(s). Each task uses either a front ZED2 camera or wrist cameras mounted near the grippers with RGB observations. We describe the tasks details and the success criteria below.

- *Box Lift*: A bi-manual task where two arms lift a box and balance in the air for at least two seconds, using the front camera and external joint torque from the arms.
- *Non-Prehensile Pivot*: The robot flips an item by pivoting it against the corner of a fixture until the item is rotated by 90° and can stand stably, using the front camera and external joint torque from the arm.
- *Fruit Pick and Place*: The robot grasps a soft and delicate fruit and places it in a bowl, using the wrist camera and the external joint torque of the gripper.

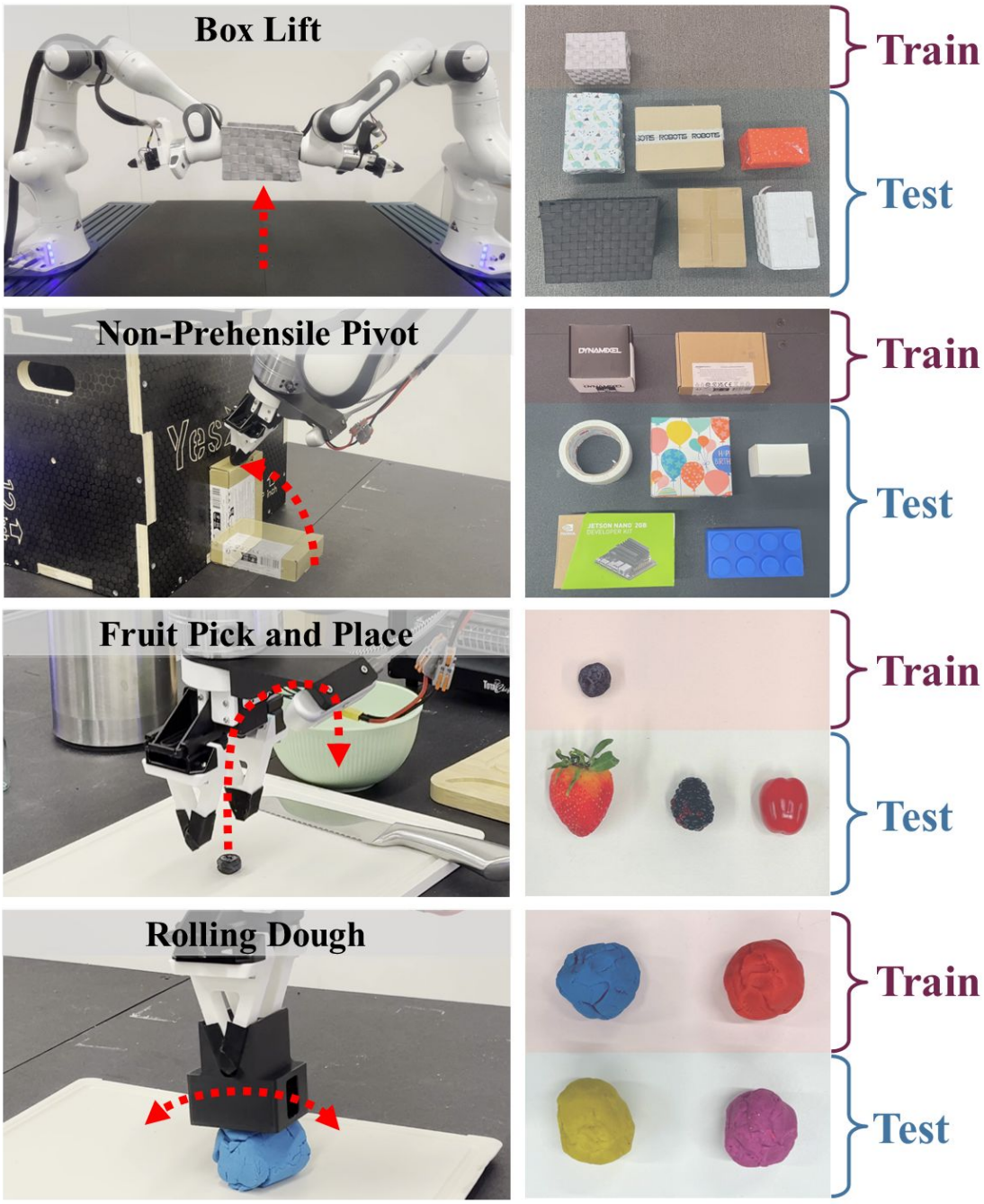


Figure 3.5: **Tasks.** We evaluate our leader-follower teleoperation system and autonomous policies trained with FACTR on four contact-rich tasks. These tasks are challenging as they require the robot to perceive and respond to the force feedback as it manipulates objects with unseen visual appearances and geometries.

- *Rolling Dough*: The robot continuously rolls the dough to shape it into a cylinder for at least 8 seconds, using the front camera and the external joint torque of the arm.

3.4.2 Teleoperation Evaluation

We compare our leader-follower teleoperation system, which includes our leader arm with force feedback, gravity compensation, and redundancy resolution, to an un-actuated leader-follower baseline system with mechanical joint regulation, similar to [129]. We summarize our results in Fig. 3.7.

Our experiments show that our system allows users to complete tasks with 64.7% higher task completion rate, 37.4% reduced completion time, and 83.3% improvement in the subjective ease of use metrics.

We observe that for tasks that require continuous contact between the arm and an object, such as non-prehensile pivoting and bimanual box lifting, the un-actuated teleoperation system often causes the follower arm to lose contact with the object. This occurs because of the absence of force feedback, which prevents the user from perceiving the environment's geometric constraints through the leader arms. As a result, maintaining continuous contact with the object becomes challenging.

For the un-actuated system, the follower arm frequently exceeds its joint velocity limits when moving under continuous contact. This occurs because the operator can easily maneuver the leader arms in ways that cause significant deviations between the leader and follower joint positions, especially when the follower arm is in contact with the environment. When contact is lost, the resulting large joint-space error causes the PID controller to generate large torques, causing abrupt movements that exceed the velocity limits. On the other hand, our system's force feedback renders geometric constraints of the environment for the operator through the leader arms, preventing the operator from moving the leader arms too far away from the follower arms during environment contacts.

3.4.3 Policy Evaluation

Questions. In our real-world evaluation, we seek to address the following research questions regarding FACTR:

- How does FACTR perform compared to baseline approaches that do not use force feedback and ones that use force feedback without FACTR?
- How do different curriculum parameters affect policy performance?

Training and Evaluation Protocol. We collected 50 demonstrations with our teleoperation system. We trained each method with the same hyperparameters, where details can be found in the Appendix B.4. We compare the following methods:

- *ACT (Vision-Only) [141]*: Action Chunking Transformer which only takes in visual observation.
- *ACT (Vision+Force) [53, 61]*: Action Chunking Transformer which takes in both visual and force observation, but trained without a curriculum.

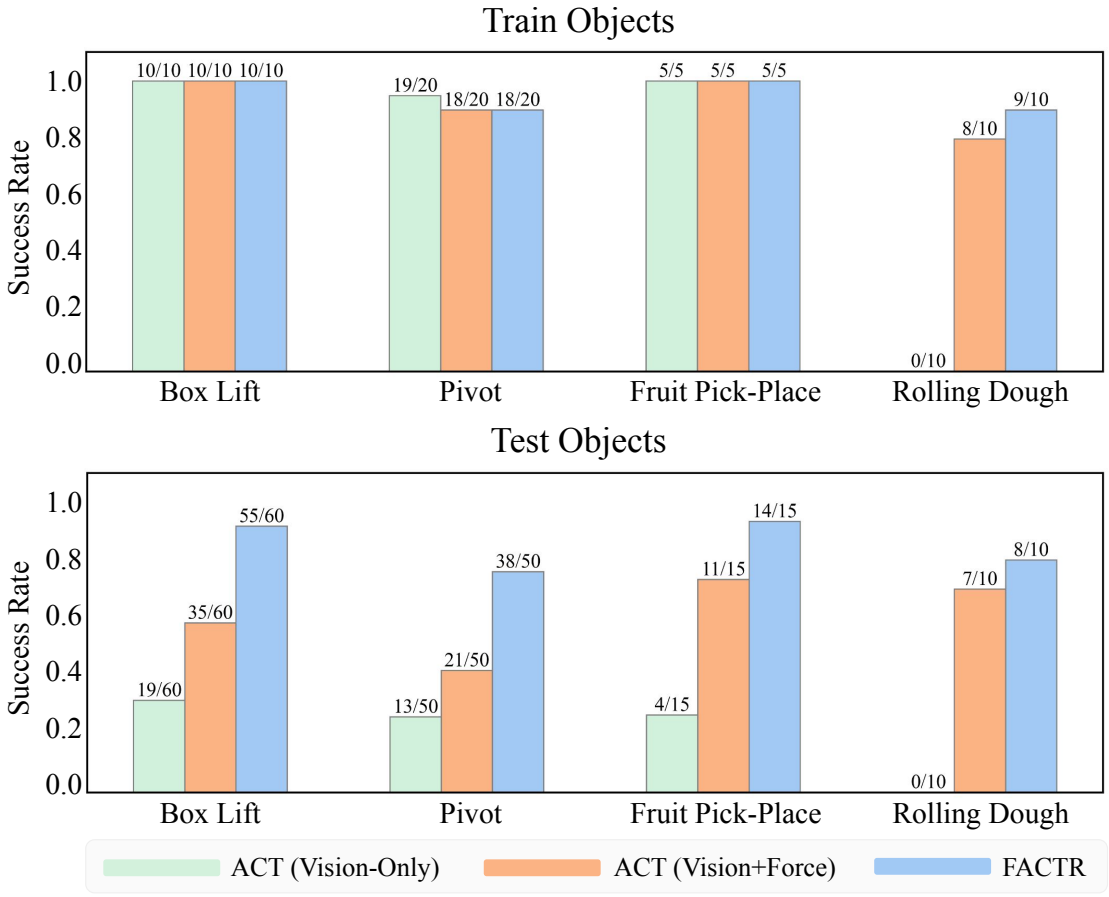


Figure 3.6: FACTR leads to better object generalization.

- *FACTR (Ours)*: Action Chunking Transformer trained with Force-Attending Curriculum. For each task, we train a latent space curriculum with the Guassian Blur operator and linear scheduler. We discuss more detailed ablations on the curriculum in Sec. 3.4.4.

For each object in each task, we evaluated 5-10 trials. We present the average success rate for training and testing objects, respectively. Detailed evaluation results for each object can be found in the Appendix B.6.

FACTR leads to better generalization. We present our main quantitative results in Fig. 3.6. All the policies perform similarly on the train objects for most tasks, except for the rolling dough task, where the vision-only policy smashes the dough without any rolling actions and fails completely. Note that the visual observations are hard to distinguish during the oscillatory rolling motions, while the force signals form a corresponding oscillatory pattern, as shown in Fig. 3.8; this distinctive torque pattern helps policies with force input to complete the task.

For the test objects, the vision-only policy achieves a success rate of 21.3% on average, which is significantly worse than policies incorporating force. Without a curriculum, policies naively incorporating force achieve a success rate of 61.2%, while FACTR achieves a success rate of 87.5%, which shows that FACTR leads to significantly better generalization to novel objects. We hypothesize that the force information provides important signals for mode switching at moments such as when the robots get into contact with the box in the lifting task and when the object is grasped in the fruit pickup task.

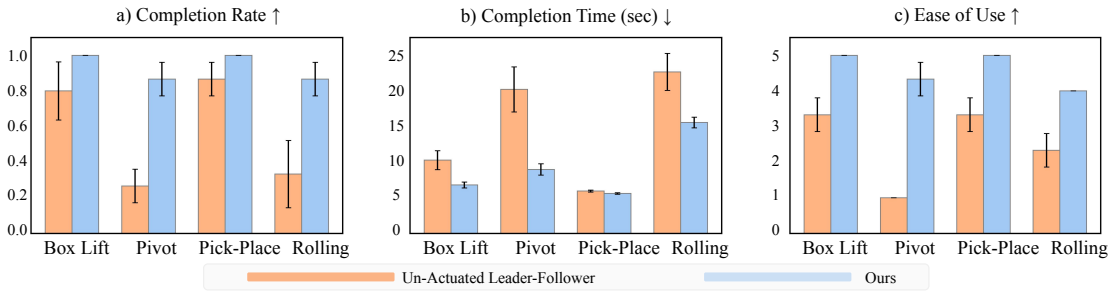


Figure 3.7: **User study.** FACTR teleoperation system allows users to complete tasks with significantly higher success rate, using less time, and they subjectively found our system to be easier to use.

	Train Objects	Test Objects
ACT (Vision-Only)	4/5	4/30
ACT (Vision+Force)	5/5	16/30
FACTR	**3/3**	**27/30**

Table 3.1: Evaluation of recovery behaviors for box lifting.

Policies with FACTR learns to identify mode switching. To better understand the policies trained with FACTR. We visualize the attention behavior during policy training and inference. Specifically, we visualize the cross attention of the action tokens to the memory tokens denoted as $\alpha_V^{(1)}$ and $\alpha_F^{(1)}$ for the first layer of the decoder, where $\alpha_V^{(1)}$ and $\alpha_F^{(1)}$ are defined in Sec. 3.3.

During policy rollout, we visualize the average cross attention of the action tokens to the force or vision tokens of the first decoder layer as shown in Fig. 3.9. FACTR learns to attend to force more during task execution. For example, in the box lifting task, attention to force outweighs that of vision as the arms contact the box, signaling a mode switch. While without the curriculum, the policy does not pay enough attention to force, and either fails to lift or balance the novel boxes.

FACTR leads to better recovery behavior. Another notable observation is that FACTR also facilitates recovery behavior. Specifically, we evaluate the box-lifting task with five trials per object. A trial begins when the policy successfully lifts the box for the first time; we then knock the box down and assess the second attempt. As shown in Table 3.1, all policies maintain nearly 100% recovery success on training objects. However, for test objects, the vision-only policy's success rate drops significantly from 31.7% on the first attempt to 13.3% on the second. In contrast, force-attending policies maintain similar success rates across both attempts.

We observe that vision-only policies often remain static after the box is knocked down, failing to retry. We hypothesize that this occurs because the vision-only policy overfits to training scenarios, making it unresponsive to unseen objects outside its training distribution. In contrast, FACTR policies detect loss of contact through external joint torque readings, which revert to pre-lift values when the object is dropped. Since our FACTR policies effectively attend to force input, they successfully recover to a pre-lift state and attempt the task again.

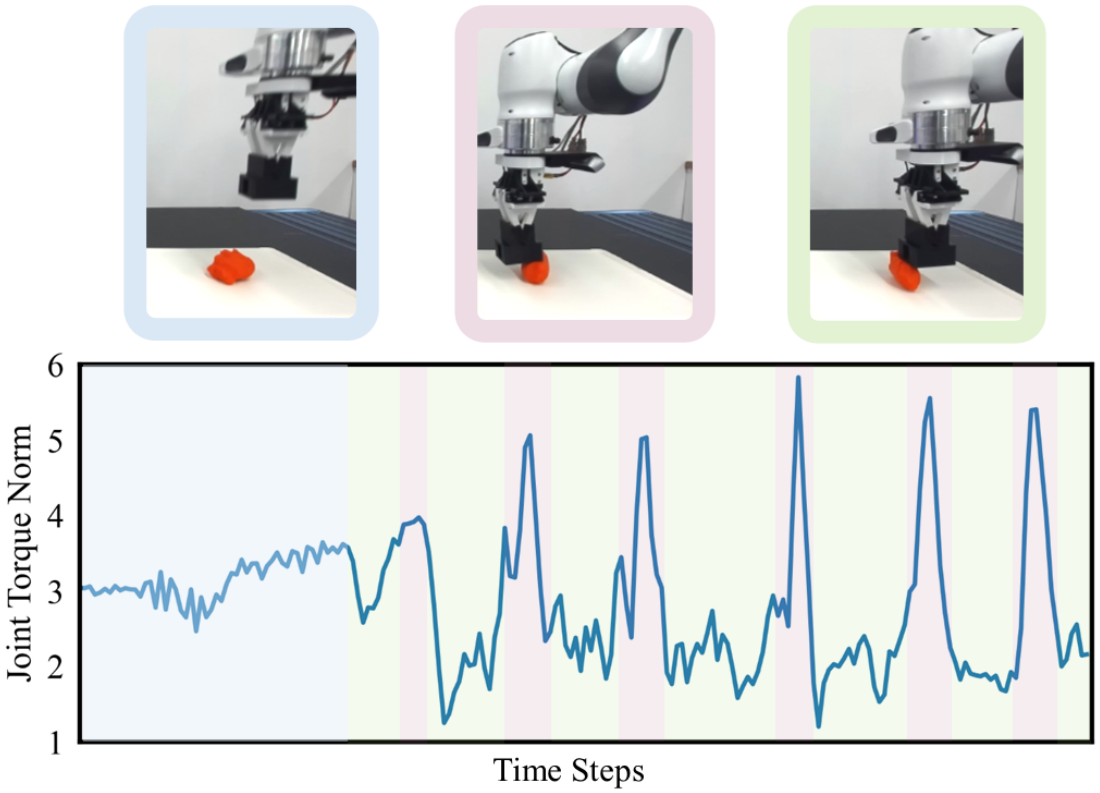


Figure 3.8: We visualize the external joint torque norm of the Franka arm for a collected trajectory. Blue highlights indicate pre-contact phases, while purple and green mark torque peaks and troughs at the dough's left and right ends, respectively. The oscillatory torque pattern helps the policy distinguish observations despite similar visual inputs.

3.4.4 Ablations on Curriculum

To further validate the significance of a curriculum, we trained models with fixed σ_n across training. Moreover, to ablate on pixel space and latent space curriculum, and different scheduler and operator choices, we train policies on different combination of these parameters. We choose the task of *pivoting*, one of the hardest tasks from our task suite, for the ablations. We evaluate only on the five *test objects* for five trials each, since they are more indicative of policy performance than train objects. The results are presented in TABLE 3.2.

	Pixel Space		Latent Space	
	Blur	Downsample	Blur	Downsample
Constant	16/25	15/25	17/25	16/25
Linear	19/25	18/25	19/25	18/25
Cosine	20/25	19/25	17/25	19/25
Exp	19/25	21/25	20/25	19/25
Step	19/25	18/25	20/25	19/25

Table 3.2: **Curriculum ablation.**

Fixed-Scale Operator vs. Curriculum. We found that performance with a curricu-

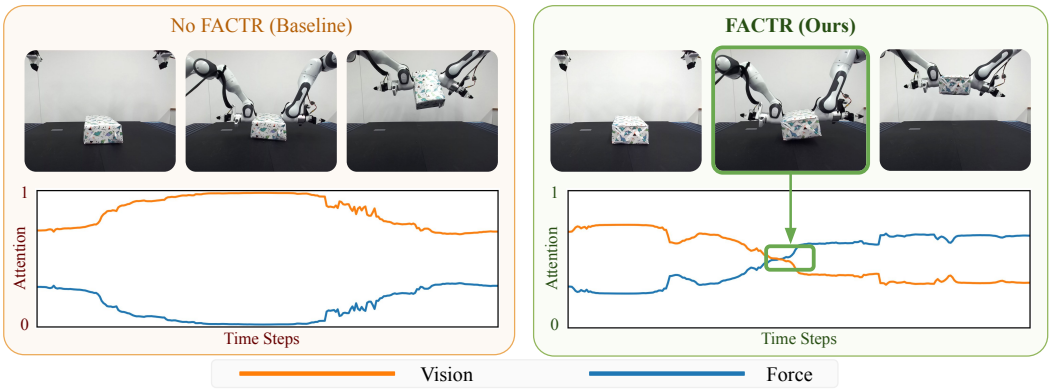


Figure 3.9: **Policies trained with FACTR learns to identify mode switching.** We visualize the average cross attention of the action tokens to the force or vision tokens of the first decoder layer during policy rollout. [Left] Without the curriculum, the policy does not pay enough attention to force, and either fails to lift or balance the novel boxes. [Right] FACTR learns to attend to force more to complete the task. For example, in the box lifting task, attention to force outweighs that of vision as the arms contact the box, signaling a mode switch.

lum of decaying smoothing performs better than a fixed curriculum across all tasks. We hypothesize that to enable better performance, the final policy needs to take in the fully unblurred vision information. Through the curriculum, a policy gets to gradually adapt to unblurred images. On the other hand, with fixed smoothing, even though policy may not overfit to visual information, it cannot extract the necessary details from unblurred vision to complete the tasks.

Comparisons with other scheduler parameters. We further compare policies trained with either pixel space or latent space, two operators (Gaussian blur and downsample) as defined in Sec. 3.3.3, and four schedulers (linear, cosine, exponential, and step) as defined in Sec. 3.3.4. However, we do not find a uniform advantage or disadvantage for any set of parameters. The results suggest that FACTR is relatively robust to different sets of curriculum parameters.

3.5 Discussion

We introduced FACTR, a curriculum approach to train force-based policies to improve performance and object generalization in contact-rich tasks. FACTR leverages a blurring operator with decreasing scales on the visual information throughout training. This encourages the policy to leverage force input at the beginning stages of training, preventing the problem where the policy overfits to visual input and thus neglects force input. This approach was demonstrated through a series of experiments on the following tasks: box lifting, non-prehensile pivoting, fruit pick-and-place, and rolling dough, where FACTR exhibits significant improvements in task completion rates and generalization to unseen object appearances and geometries. Additionally, our teleoperation system, which includes an actuated leader arm for force feedback and gravity compensation, was shown to provide a more intuitive user experience, as evidenced by higher task completion rates and user satisfaction in our studies.

While FACTR demonstrates significant improvements in force-based policy learning for contact-rich tasks, it has limitations. First, the precision of the external joint torque sensors in our follower arm is limited. This limitation can particularly affect tasks

that involve subtle force adjustments during fine-grained manipulation since the torque readings can be too noisy to be used effectively. Future work could explore integrating high-resolution tactile sensors or haptic gloves to enhance feedback precision and improve overall system performance. Second, our approach assumes the availability of external joint torque sensors in the follower arms. Future work can explore adapting our system for an arm mounted with an end-effector force-torque sensor. Third, the effectiveness of our curriculum learning approach can be influenced by several hyperparameters, such as the choice of the blurring operator and scheduling strategies. These parameters can be highly task-dependent, requiring extensive tuning for different applications. Developing adaptive or self-tuning curriculum strategies could help mitigate this issue by dynamically adjusting hyperparameters based on task-specific requirements. Addressing these limitations could further enhance FACTR's applicability and robustness across a broader range of contact-rich manipulation tasks.

Chapter 4

Deep Reactive Policy: Learning Motion Generation from Experts

To operate in natural human environments like homes and kitchens, robots must navigate safely in fast-changing, partially observable settings. This demands an intuitive understanding of robot's own physical presence within the world. At the core of this capability is collision-free motion generation. Motion generation can operate beneath high-level policy layers such as VLMs or behavior cloning agents, ensuring that the robot's actions remain both safe and physically feasible.

To achieve this, robots have traditionally relied on motion planning approaches. Search-based methods, such as A* [35] and AIT* [115], are capable of finding globally optimum solutions, but they assume complete knowledge of the environment and static conditions. Due to their long execution times, these planners are typically run offline to generate fixed open-loop trajectories that the robot executes, limiting their performance in avoiding dynamic obstacles. Reactive controller-based approaches [49, 92, 119, 6], such as Riemannian Motion Policies (RMP) [92] and Geometric Fabrics [119], offer reactive collision avoidance in dynamic scenes. However, these approaches lack global scene awareness and often become trapped in local minima in complex environments [29].

An alternative is to formulate motion generation as a visuo-motor neural policy that maps raw visual observations directly to actions. Unlike open-loop planners, a learned visuo-motor policy continuously processes live sensory inputs—such as point clouds—to adapt its behavior on the fly without requiring known models of the scene. This real-time closed loop adaptability is crucial in scenarios where the goal becomes temporarily obstructed by dynamic obstacles or during sudden environmental changes.

Generating such visuo-motor neural policies requires a robust training methodology. Several works have proposed using traditional motion planners to produce ground truth trajectories to serve as supervision for training [90, 29, 21, 30]. Unfortunately, prior methods such as MπNet [29] have only demonstrated limited success in simple settings and often fail to generalize to unseen environments, constraining their broader applicability. More recent efforts, like NeuralMP [21], attempt to address these generalization issues by introducing test-time optimization to correct for policy inaccuracies. While this improves accuracy, it requires runtime search before execution, sacrificing the reactivity necessary for fast-changing environments.

Our method, Deep Reactive Policy (DRP), is a visuo-motor neural motion policy designed for reactive motion generation in diverse dynamic environments, operating

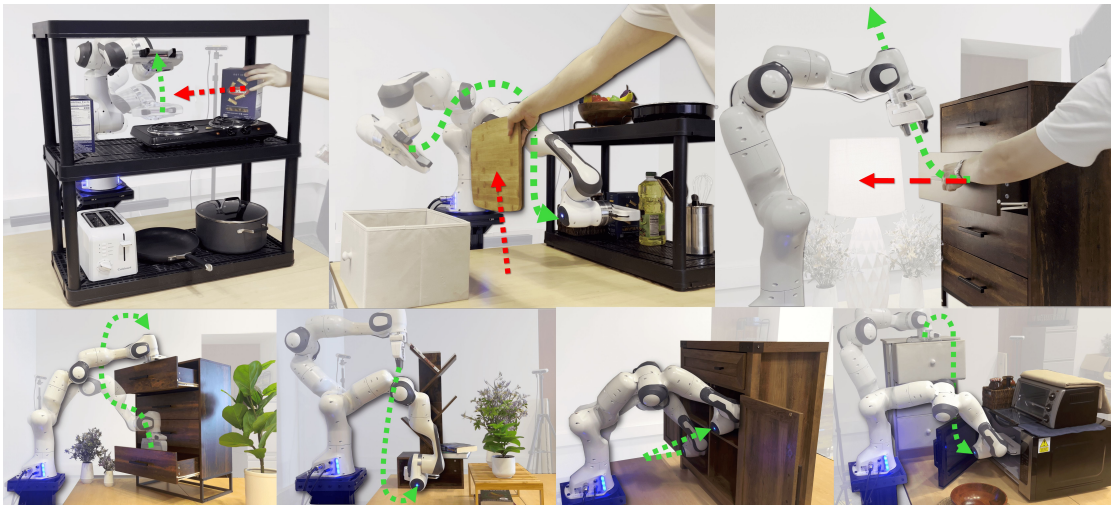


Figure 4.1: We present Deep Reactive Policy (DRP), a point cloud conditioned motion policy capable of performing reactive, collision-free goal reaching in diverse complex and dynamic environments.

directly from point cloud sensory input. At the core of DRP is IMPACT (Imitating Motion Planning with Action-Chunking Transformer), a transformer-based neural motion policy. First, we pretrain IMPACT on 10 million motion trajectories generated in diverse simulation environments leveraging cuRobo [116], a state-of-the-art GPU-accelerated motion planner. Next, we finetune IMPACT with an iterative student-teacher method to improve the policy's static obstacle avoidance capabilities. Finally, we integrate a locally-reactive goal-proposal module, DCP-RMP, with IMPACT to further enhance dynamic reactivity. Altogether, we call our approach Deep Reactive Policy (DRP), enabling the policy to generalize from learned experiences and develop an intuitive understanding of the changing environment.

Our core contributions are:

- We scale up motion data generation to train IMPACT, a novel end-to-end transformer-based neural motion policy conditioned on point cloud observations.
- We further improve IMPACT's obstacle avoidance performance via finetuning with iterative student-teacher distillation.
- We enhance IMPACT's dynamic obstacle avoidance performance via a locally reactive goal-proposal module, DCP-RMP.

We evaluate DRP on both simulation and real-world environments, featuring complex obstacle arrangements and dynamic obstacles. DRP consistently outperforms previous state-of-the-art motion planning methods, as detailed in Section 4.3.

4.1 Related Work

Our work is inspired by the following related work:

Global Planning Methods Global planners generate collision-free trajectories by exploring the full state space, offering asymptotic completeness and optimality. Search-based methods like A* [35] and its variants [64, 63, 54] guarantee optimality under

admissible heuristics but scale poorly in high-dimensional continuous domains due to discretization. Sampling-based planners such as PRM [46], RRT [58], and their extensions [7, 59, 55, 33, 115] improve scalability and efficiency through continuous-space sampling and informed exploration. Trajectory optimization methods [145, 100, 27] refine trajectories via continuous optimization but are sensitive to initialization and local minima. Recent work, cuRobo, advances trajectory optimization with GPU parallelization. However, it still plans from scratch for each new problem and relies on accurate collision checking, limiting its applicability in real-world and dynamic environments. In contrast, our method overcomes these challenges by learning from diverse planning data and directly generating actions from raw point cloud inputs.

Locally-Reactive Methods Locally reactive controllers generate collision-free motion by steering toward goals while avoiding nearby obstacles [49, 92, 119, 6]. Though effective in dynamic settings, they often get trapped in local minima within cluttered environments [29]. We address this by training a neural policy on globally aware planners, enabling it to produce globally feasible motion even in complex scenes while retaining closed-loop reactivity.

Learning-based Methods Neural networks run efficiently at inference time, and have been widely used to accelerate motion planning. A line of work augments classical motion planners with learned components that bias sampling [90, 57, 140, 15], guide search [42, 89] or initialize optimization [39]. These hybrid approaches preserve the robustness of classical planning while leveraging learning to improve sample efficiency and planning speed, particularly in high-dimensional or constrained scenarios. In contrast, more recent methods [44, 14, 99] generate feasible motion trajectories directly, handling dynamic constraints, capturing multimodal distributions, and incorporating scene-specific costs without explicit planning at inference.

While effective, these methods often rely on ground-truth states, known obstacle geometry, or open-loop trajectory prediction without closed-loop reactivity. To overcome these limitations, recent approaches develop end-to-end policies that operate directly on point clouds, facilitating real-world deployment without the need to perform scene reconstruction. MπNets [29] and MπFormer [30] use supervised learning on point cloud inputs, achieving strong in-distribution results, but struggling to generalize due to limited training diversity. NeuralMP [21] improves generalization through scene and obstacle randomization, but relies on slow test-time optimization, limiting real-time performance. Our method addresses both issues, providing robust generalization and fast closed-loop execution.

4.2 Deep Reactive Policy (DRP)

We present Deep Reactive Policy (DRP), a neural visuo-motor policy that enables collision-free goal reaching in diverse, dynamic real-world environments. An overview of the full system architecture is shown in Figure 4.2.

At the core of DRP is IMPACT, a transformer-based policy that generates joint position targets conditioned on a joint-space goal and live point-cloud input. IMPACT is trained in two phases. First, it undergoes pretraining via behavior cloning on a large offline dataset with over 10M trajectories generated by cuRobo [116], a state-of-the-art optimization-based motion planner. While this pretraining demonstrates strong global planning potential, the resulting policy often incurs minor collisions, as the kinematic

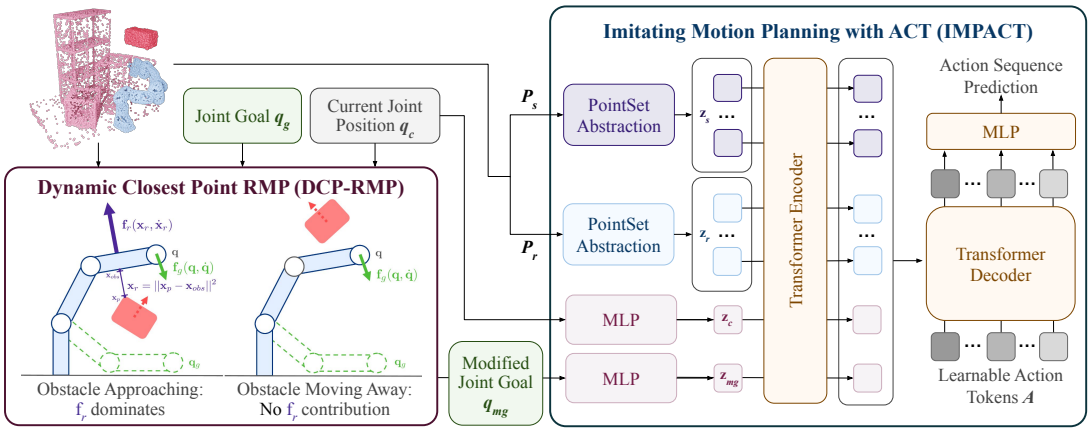


Figure 4.2: Deep Reactive Policy (DRP) is a visuo-motor neural motion policy designed for dynamic, real-world environments. First, the locally reactive DCP-RMP module adjusts joint goals to handle fast-moving dynamic obstacles in the local scene. Then, IMPACT, a transformer-based closed-loop motion planning policy, takes as input the scene point cloud, the modified joint goal, and the current robot joint position to output action sequences for real-time execution on the robot.

expert trajectories neglect robot dynamics.

Subsequently, we enhance IMPACT's static obstacle avoidance using student-teacher finetuning. The teacher combines the pretrained IMPACT policy with Geometric Fabrics [119], a state-based closed-loop controller that excels at local obstacle avoidance while respecting robot dynamics. Since Geometric Fabrics relies on privileged obstacle information, we distill its behavior into a fine-tuned IMPACT policy that operates directly on point-cloud inputs.

To further boost reactive performance to dynamic obstacles during deployment, DRP utilizes a locally-reactive goal proposal module, DCP-RMP, that supplies real-time obstacle avoidance targets to the fine-tuned IMPACT policy.

4.2.1 Large-Scale Motion Pretraining.

To enable supervised pretraining of a motion policy that can learn general collision-free behavior, we generate diverse and complex training scenes paired with expert trajectory solutions. While we largely follow the data generation pipeline introduced in [21], we replace AIT* with cuRobo as the expert motion planner. Due to its GPU acceleration, cuRobo allows us to scale data generation to 10 million expert trajectories.

We also introduce challenging scenarios where the goal itself is obstructed by the environment and thus physically unreachable. In these cases, we modify the expert trajectory to stop the robot as close as possible to the goal without colliding with the blocking obstacle. This scenario is critical to include, as in dynamic settings, obstacles like humans may temporarily block the target, and the robot must learn to avoid collisions even when the goal cannot be immediately reached.

We then train with this data using **IMPACT** (Imitating Motion Planning with Action-Chunking Transformer), a transformer-based neural motion policy architecture. IMPACT outputs joint position targets, conditioned on the obstacle point cloud $P_s \in \mathbb{R}^{N_s \times 3}$, robot point cloud $P_r \in \mathbb{R}^{N_r \times 3}$, current joint configuration $q_c \in \mathbb{R}^7$, and goal joint configuration $q_{mg} \in \mathbb{R}^7$.

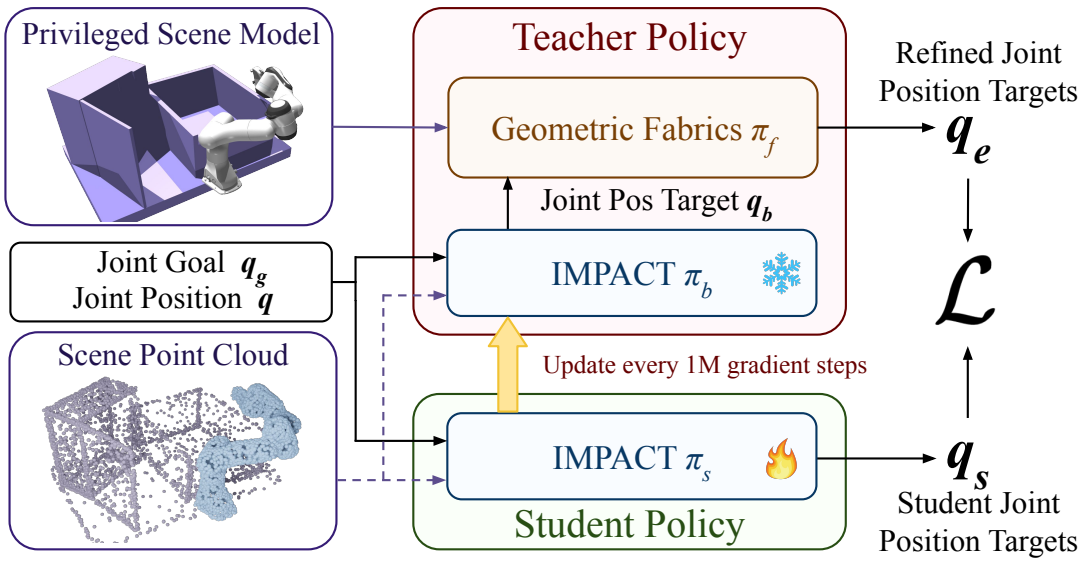


Figure 4.3: The IMPACT policy is combined with a locally reactive Geometric Fabrics controller to enable improved obstacle avoidance. This combined teacher policy is then distilled into a point-cloud conditioned student policy.

During training, point cloud inputs are generated by uniformly sampling points from ground-truth mesh surfaces in simulation. During real-world deployment, scene point clouds are captured using calibrated depth cameras. We also replace points near the robot in the captured point cloud with points sampled from its mesh model, using the current joint configuration to ensure an accurate representation of its state.

To reduce computational complexity and enable real-time inference, we use set abstraction from PointNet++ [87] to downsample the point clouds and generate a smaller set of latent tokens. Specifically, the scene and robot point clouds are converted into tokens $z_s \in \mathbb{R}^{K_s \times H}$ and $z_r \in \mathbb{R}^{K_r \times H}$, where $K_s < N_s$ and $K_r < N_r$. The current and target joint angles are encoded using MLPs to produce $z_c, z_{mg} \in \mathbb{R}^H$, respectively. Each input is paired with a learnable embedding: $e_s, e_r, e_c, e_{mg} \in \mathbb{R}^H$, which are added to the corresponding tokens to form the encoder input.

The decoder processes S learnable action tokens $A \in \mathbb{R}^{S \times H}$, using the encoder output as memory. The decoder outputs a sequence of S delta joint actions $[\bar{q}_1, \bar{q}_2, \ldots, \bar{q}_S] \in \mathbb{R}^{S \times 7}$, which are supervised using a Mean Squared Error (MSE) loss against the ground-truth actions $[q_1, q_2, \ldots, q_S]$:

$$\mathscr{L}_{BC} = \frac{1}{S} \sum_{i=1}^{S} ||q_i - \bar{q}_i||_2$$

These delta actions are then converted into absolute joint targets and sent to the robot's low-level controller for real-time execution.

4.2.2 Iterative Student-Teacher Finetuning

Pretraining IMPACT on our dataset provides a strong globally-aware motion policy, already outperforming prior state-of-the-art neural methods (see Section 4.3.1). However, since the expert trajectories generated offline from cuRobo are purely kinematic, they

fail to capture robot dynamics and controller behavior, resulting in frequent collisions at deployment.

To enhance the policy's ability to understand robot dynamics and further improve static obstacle avoidance, we improve IMPACT via iterative student-teacher finetuning. Since many of the pretrained policy's failure cases stem from minor collisions with local obstacles, we can remedy these mistakes by passing IMPACT's joint position outputs into Geometric Fabrics [119], a state-of-the-art controller that excels at local obstacle avoidance.

Geometric Fabrics uses ground-truth obstacle models to follow joint targets while avoiding nearby obstacles and respecting dynamic constraints like joint jerk and acceleration limits. Since it relies on privileged information, we apply student-teacher distillation in simulation to refine our point-cloud-based IMPACT policy [98].

We initialize the student policy π_s with the pretrained IMPACT policy from Section 4.2.1. The teacher policy also starts with the pretrained IMPACT policy π_b, which outputs an action chunk of joint position targets. We take the first action in this chunk, q_b, as an intermediate goal and pass it to Geometric Fabrics π_f, which refines it into improved targets $q_e = \pi_f(obs, q_b)$. These refined targets then supervise updates to the student policy π_s. To ensure scalability, the entire process runs in parallel using IsaacGym [71]. Since Geometric Fabrics requires signed distance fields (SDFs) for obstacle avoidance, we precompute them offline in batch for static scenes. Notably, we avoid using cuRobo as the expert during finetuning, as it would require planning at every simulation step across all vectorized environments—rendering it computationally impractical.

During distillation, we keep π_b frozen within the teacher policy to maintain stable objectives. After one million gradient steps, the fine-tuned student replaces π_b, and the process repeats. This iterative procedure progressively improves local obstacle avoidance while preserving strong global planning capabilities. The full process is illustrated in Figure 4.3. This iterative student-teacher finetuning improves the success rate over the pretrained model by 45%, as shown in Table 4.1.

4.2.3 Riemannian Motion Policy with Dynamic Closest Point

While IMPACT's closed-loop nature allows it to implicitly handle dynamic environments—making decisions at every timestep—its performance degrades in particularly challenging scenarios, such as when an object moves rapidly toward the robot. This limitation arises because dynamic interactions are absent from both the pretraining dataset and the student-teacher finetuning phase, as generating expert trajectories for non-static scenes would require re-planning after every action, which is computationally infeasible for both cuRobo and our vectorized Geometric Fabrics implementation.

To explicitly enhance IMPACT's dynamic obstacle avoidance during inference, we introduce a Riemannian Motion Policy (RMP) layer. This non-learning based component uses local obstacle information to enable highly reactive local obstacle avoidance. Specifically, RMP acts as a goal-proposal module, modifying the original joint-space goal $\mathbf{q}_g$ into a new goal $\mathbf{q}_{mg}$ that prioritizes avoidance when dynamic obstacles approach. This adjusted goal is then passed to IMPACT. Notably, because IMPACT is already trained with global scene awareness, focusing RMP solely on local dynamic obstacles does not compromise the global goal-reaching performance.

However, RMP traditionally requires ground-truth obstacle models and poses to generate joint targets for reactive avoidance, limiting its real-world deployability. To

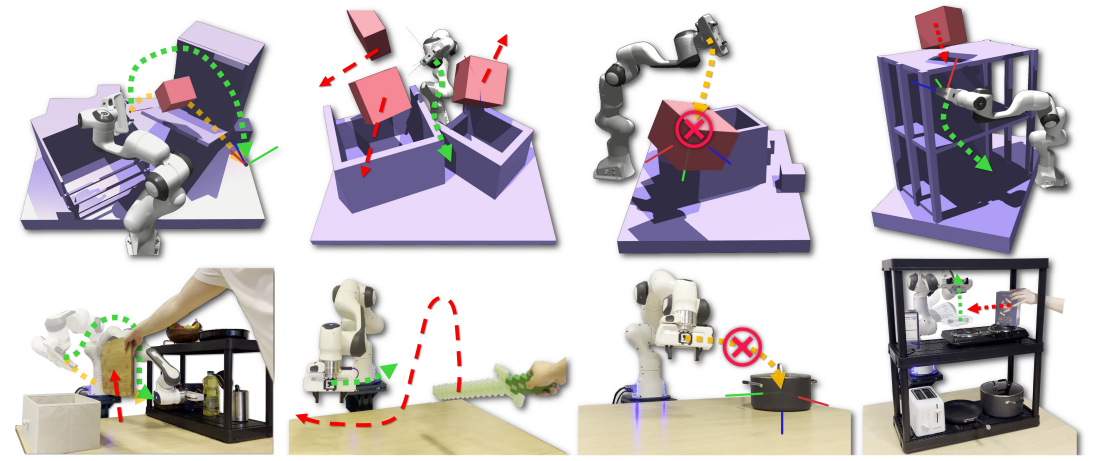


Figure 4.4: **DRPBench** introduces five challenging evaluation scenarios in simulation and real. In addition to complex Static Environments **(SE)**, we propose Suddenly Appearing Obstacle **(SAO)** where obstacles appear suddenly ahead of the robot, Floating Dynamic Obstacles **(FDO)** where obstacles move randomly throughout the environment, Goal Blocking **(GB)** where the goal is obstructed and the robot must approach as closely as possible without colliding, and Dynamic Goal Blocking **(DGB)** where the robot encounters a moving obstacle *after* getting to the goal.

overcome this, we propose Dynamic Closest Point RMP (DCP-RMP)—an RMP variant that operates directly on point cloud inputs. At a high level, DCP-RMP identifies the closest point in the point cloud belonging to a dynamic obstacle and generates repulsive motion to steer the robot away.

Specifically, we implement DCP-RMP by first extracting the dynamic obstacle point cloud using a KDTree, which efficiently performs nearest-neighbor queries between the current and previous frame point clouds to identify moving points. We then compute the minimal displacement $\mathbf{x}_r$ between the robot and nearby dynamic obstacles and derive a repulsive acceleration to increase this separation. Finally, we adjust the original joint goal $\mathbf{q}_d$ by virtually applying this repulsive signal, yielding the modified goal $\mathbf{q}_{mg}$ that prioritizes dynamic obstacle avoidance. Detailed mathematical formulations are provided in the Appendix.

While the modified joint goal may sometimes intersect with static obstacles, rendering it physically unachievable, IMPACT has been trained on scenarios where the goal configuration is in collision with the scene and learns to stop safely in front of obstacles instead.

4.3 Evaluation

To comprehensively evaluate DRP's reactivity and robustness, we design **DRPBench**, a set of challenging benchmark tasks in both simulation and the real world. These tasks target three critical real-world challenges for robot motion generation: navigating cluttered static environments, reacting swiftly to dynamic obstacles, and handling temporarily obstructed goals with caution and precision.

We report quantitative results and address the following key questions: (1) How does

	DRPBench					MπNets Dataset
	SE	SAO	FDO	GB	DGB	
With privileged information:						
AIT* [115]	40.50	0	0	0	0	89.22
cuRobo [116]	82.97	59.00	39.50	0	3.00	**99.78**
cuRobo-Vox [116]	50.53	58.67	40.50	0	2.50	-
RMP [92]	32.97	46.0	49.50	**71.08**	50.50	41.90
MπNets [29]	2.50	0.33	0	0	0	65.18
MπFormer [30]	0.19	0	0	0	0	30.02
NeuralMP [21]	50.59	33.16	19.00	0	0.25	–
IMPACT (no finetune)	58.25	47.33	13.50	46.50	0	66.27
IMPACT	**84.60**	**86.00**	32.00	66.67	0.25	83.71
DRP (Ours)	**84.60**	**86.00**	**75.50**	66.67	**65.25**	83.71

Table 4.1: Quantitative results on simulation DRPBench and MπNets Dataset. DRP outperforms all classical and learning-based baselines across diverse settings, particularly excelling in dynamic and goal-blocking tasks. While optimization methods like cuRobo succeed in static scenes, they struggle under dynamic conditions. DRP's combination of architectural improvements, fine-tuning, and reactive control (via DCP-RMP) enables robust generalization and superior performance, especially in scenarios requiring fast adaptation.

DRP perform compared to state-of-the-art classical and learning-based motion planners? (2) What are the individual contributions of DRP's architectural design, student-teacher fine-tuning, and DCP-RMP integration? (3) Can DRP generalize effectively to real-world environments, especially those exhibiting significant domain shift from training?

4.3.1 Simulation Experiments and Results

For the simulation experiments, **DRPBench** comprises over 4000 diverse problem instances, with examples shown in Figure 4.4. Further details about tasks are provided in the Appendix. In addition to DRPBench, we also test on the MπNets benchmark [29] in a zero-shot manner without additional training. We use success rate as the primary metric, where a trial is successful if the robot reaches the end-effector goal pose within position and orientation thresholds without collisions.

Classical methods fail in dynamic and goal-blocking scenes. Sampling-based planners such as AIT* completely fail in dynamic environments, achieving 0% success on all such tasks despite extended planning horizons. Optimization-based approaches like cuRobo and RMP perform significantly better in static settings—e.g., 82.97% and 32.97% on Static Environment (SE), respectively—but degrade in harder scenarios. cuRobo drops to 3.00% on Dynamic Goal Blocking (DGB), where the goal is temporarily obstructed. In contrast, RMP achieves 50.50% on DGB, motivating its integration into DRP as a reactive control module. Still, DRP surpasses RMP significantly on tasks except for Goal Blocking (GB), underscoring the benefit of combining learning with reactive control.

Architectural design and training diversity enables generalization. Learning-based models trained on narrow datasets—such as MπNets and MπFormer—fail to

	FDO		DGB	
Has DCP-RMP?	✗	✓	✗	✓
cuRobo [116]	39.50	51.50	3.00	54.50
NeuralMP [21]	19.00	34.00	0.25	20.75
IMPACT	32.00	**75.50**	0.25	**65.25**

Table 4.2: DCP-RMP is method-agnostic.

	SE	SAO	FDO	GB	DGB
cuRobo-Vox	60.00	3.33	0	0	0
NeuralMP [21]	30.00	6.67	0	0	0
IMPACT	**90.00**	**100.00**	0	**92.86**	0
DRP (Ours)	**90.00**	**100.00**	**70.00**	**92.86**	**93.33**

Table 4.3: Success Rate (%) on Real-world tasks.

generalize to out-of-distribution scenes, achieving only 0–2.5% on the DRPBench tasks. NeuralMP performs better, but it relies on test-time optimization (TTO)—a process that adapts trajectories post-hoc during deployment which only helps in Static Environment (SE) and struggles in reactive contexts. However, even without finetuning, IMPACT outperforms other learning-based methods due to its more expressive, scalable architecture and training on a more diverse dataset—without relying on post-hoc techniques.

Fine-tuning surpasses data generation method. DRP is pretrained using trajectories from a classical planner (cuRobo), but it significantly exceeds this source's performance. This is because we filter for successfully planned trajectories during data generation and we apply a student-teacher fine-tuning stage that distills and refines action generation beyond the original data. The result is a policy that not only inherits useful behaviors but generalizes more effectively across complex settings. IMPACT also performs well in static and dynamic environments that require fine local control. It achieves 86.00% on Suddenly Appearing Obstacle (SAO) and 66.67% on Goal Blocking (GB), where precise maneuvers near occluded or blocked targets are critical. These results validate the strength of closed-loop visuo-motor imitation in spatially constrained environments.

DCP-RMP boosts dynamic performance. DRP's integration of DCP-RMP adds fast local responsiveness, yielding strong results in fully dynamic tasks: 75.50% on FDO and 65.25% on DGB. In contrast, using IMPACT alone drops to 32.00% and 0.25% on the same tasks, while cuRobo reaches only 39.50% and 3.00%. These gains highlight the necessity of combining high-level learning with reactive modules to handle dynamic obstacles and shifting goals in real time. We further evaluate the impact of the DCP-RMP module by adding it to various baseline methods. As shown in Table 4.2, DCP-RMP improves performance across all dynamic tasks, regardless of the underlying method.

4.3.2 Real-World Experiment Results

Our real-world benchmark mirrors the five simulation tasks, but introduces out-of-distribution, semantically-meaningful obstacles like a slanted shelf and tall drawer, as

shown in Figure 4.1. For example, in one dynamic goal blocking (DGB) task, the robot must wait in front of a drawer, avoid a human operator, and reach in once it opens. The benchmark includes over 50 real-world instances, with two to four calibrated Intel RealSense D455 RGB-D cameras capturing the scene.

As seen in Table 4.3, DRP significantly outperforms classical and learning-based baselines in real-world settings. On tasks like Static Environment (SE) and Static Appearing Obstruction (SAO), both DRP and IMPACT achieve near-perfect success rates, far exceeding cuRobo-Vox and NeuralMP, which degrade severely under noisy perception. On the more challenging Goal Blocking (GB) task, both DRP and IMPACT reach 92.86%, while cuRobo and NeuralMP fail completely. The biggest difference appears on Floating Dynamic Obstacle (FDO) and Dynamic Goal Blocking (DGB) where IMPACT fails to solve, highlighting the value of DCP-RMP for reactive behavior. Even though being trained entirely in simulation, DRP adapts well to real-world settings.

4.4 Discussion

We introduced Deep Reactive Policy (DRP), a scalable, generalizable framework for closed-loop motion generation in complex and dynamic environments. At its core is IMPACT, a transformer-based visuo-motor policy trained on large-scale motion data and refined via student-teacher finetuning. DRP learns directly from point cloud inputs to produce collision-free, globally coherent actions while remaining robust to partial observability and environmental changes. Extensive evaluations in both simulation and real-world settings show DRP consistently outperforms prior learning-based and classical planners, especially in scenarios requiring reactive adaptation. While IMPACT addresses most planning challenges end-to-end, incorporating lightweight reactive modules like DCP-RMP further boosts performance in highly dynamic scenes. To support ongoing research, we will release all datasets, models, and benchmarks.

However, DRP is not without limitations. DRP relies on reasonably accurate point cloud observations for effective planning. Although the policy is robust to a certain degree of noise and partial observability, performance may degrade under severe perception failures. Our multi-camera setup helps mitigate this issue, but may not be suffice for tasks in narrow environments with frequent occlusions. Leveraging RGB or RGB-D inputs could improve performance for more unstructured environments.

Our experiments are also limited to a single embodiment—the Franka Panda—and we do not evaluate on other robot platforms. This limitation stems from the challenges in scaling our current pipeline to multiple embodiments. In future work, we aim to address this by either generating separate planners for each robot or training a single DRP policy that generalizes across embodiments.

Chapter 5

Object-Centric Gaussian Splatting: 3D Perception for Imitation Learning

Figure 5.1: **Object-centric Gaussian splatting.** We propose a dynamic and semantic 3D representation based on Gaussian Splatting [47], which achieves an update rate of 30 Hz in response to robot and object movements. We show the reconstruction from different viewpoints of a grasping scene on the left. We apply this representation to obtain behavior cloning policies that are robust under various testing views even though only a single training view is available. We also apply our representation to enable zero-shot language-conditioned dynamic grasping.

What representation of the scene will improve the performance and robustness of learning robots? Recent achievements in the community suggest that taking 2D RGB images as inputs allow robots to perform complex manipulation tasks [60, 18]. Nevertheless, the hidden assumption is that the camera viewpoints remain the same for training and testing. As we will demonstrate in Sec. 5.4.1, even slight shift in camera views will significantly reduce the performance of learning agents. A fixed relative pose between the cameras and the robot base or the end-effectors is an unsatisfactory requirement. As humans, we can easily solve the same tasks without our eyes fixing at a position relative to our hands. We can even easily tele-operate a robot to complete the task at completely different views. Unfortunately, most of the existing learning agents lack the 3D understanding essential to robustness of the policies.

There has been promising results on directly learning with 3D representations like voxels or point-clouds [139, 110], yet it would be optimal if learning agents can leverage immense 2D data and readily accessible pretrained vision foundation models [13, 91, 52, 81, 133]. Recent strides in integrating semantic information into neural

3D representations [48] have shown promise in enabling tasks like language-conditioned grasping [104, 109] and goal-conditioned rearrangement [125]. Yet, these approaches stumble when faced with dynamic scenes and the requirement of higher-frequency (30Hz) controls, constraining their general applicability.

The crux of the challenge lies in the resource-intensive demands of constructing semantic 3D representations which are already compute and memory-intensive for passive vision applications. Robotics adds an additional axis of time, requiring controllers at 10Hz frequency at least for practical applications. The indispensable requirement for real-time updates of the dynamic world makes 3D representation for robotics exponentially more demanding.

However, a close examination of the robotic tasks reveals a potential solution. Changes within a scene between updates are predominantly localized, suggesting that a per-step scene reconstruction may not only be inefficient but also unnecessary. By transitioning to a locally updatable scene representation, we can directly address the core of the computational challenge. This pivot from continuous, global reconstruction towards targeted, localized updates dramatically curtails the overhead associated with keeping a semantic and dynamic 3D representation, where the main computation is completed at the initialization.

Gaussian splatting [47] emerges as a promising candidate for dynamic 3D scene representation in this context. Originating from novel-view synthesis, this method employs a set of 3D Gaussian primitives to model a scene. This explicit and volumetric representation allows for local updates of the constructed scene. Further, its reliance on rasterization for rendering leverages parallel processing on GPUs, markedly accelerating rendering speeds. Nonetheless, adapting Gaussian splatting for robotics poses its own set of challenges. While it offers a speed advantage, it lacks the semantic understanding of the scene, and vitally, it still falls short of meeting the real-time update requirements for robotics.

In response to these challenges, our work builds upon static Gaussian splatting to bridge this gap. We address the need for speed and semantic interpretation by embedding "objectness" into the scene representation, thereby expediting the update process. This approach allows for rapid, high-frequency updates essential for dynamic robotic environments. This also allows a one-time extraction of 2D foundation models at the initial step for semantic information, circumventing the inference bottleneck of large models.

With our representation, we can robustify off-the-shelf 2D policy trainers to handle arbitrary camera poses by projecting observations to training views. Our semantic, dynamic, and 3D representation also allows a robot to reactively grasp moving objects prompted by open-vocabulary queries.

In summary, our contributions are:

1. Introducing the use of object-centric Gaussian splatting for dynamic, semantic, and 3D representation in robotics.

2. Overcoming the update speed limitations of the vanilla Gaussian splatting through object-centric updates, achieving 30 Hz update rate which is sufficient for most real-time robotic applications.

3. Proposing GSMimic, which utilizes our representation to obtain view-robust behavior cloning policies evaluated on simulation and real-world manipulation tasks.

4. Demonstrate the representations applicability to zero-shot language-conditioned dynamic grasping, showcasing its adaptability in dynamic settings.

5.1 Related Work

Our work is related to the following related work:

Neural Dynamic Scene Representation A pivotal advancement in neural volumetric scene representations was the introduction of Neural Radiance Fields (NeRF) [76], enabling high-quality renderings at novel views, which comes at the cost of prolonged training times. The recent development of 3D Gaussian Splatting (3D-GS) introduces a significant paradigm shift [47]. Unlike NeRF's implicit representation, 3D-GS utilizes explicit 3D Gaussian primitives, enabling scene representation, enabling fast, parallelizable rendering through rasterization. The explicit nature of 3D-GS, as opposed to the implicit form found in NeRF, has the potential for immediate updates in response to changes within the scene, making it particularly suited for dynamic environments. 3D-GS also led to several recent works that leverage the representation for offline dynamic scene reconstruction. The approaches include explicit parametrization of Gaussian parameters at different time steps and the modeling of a deformation field for Gaussians [69, 127, 135], which achieve high quality and fast rendering. These works highlight the potential for accurately capturing and rendering complex, dynamic scenes in real time. Nevertheless, they all require extensive viewpoints and offline training, while we aim at online updates with limited viewpoints for robotics applications.

3D Neural Representation for Robotic Manipulation In the exploration of 3D representations for robotic manipulation, diverse approaches have leveraged neural fields [143, 126, 41, 136]. Among these, Neural Descriptor Fields stand out for constructing neural feature fields that generalize across different instances with minimal demonstrations, yet focus primarily on geometric rather than semantic features, limiting cross-category generalization [112]. Recent efforts have distilled neural feature fields using foundation models like CLIP [91] and DINO [13, 81] for supervision. Techniques such as F3RM [109] and LERF-TOGO [48, 104] have distilled neural feature fields to facilitate language-conditioned and task-oriented grasping, demonstrating the potential of foundation models in enhancing robotic manipulation. Despite these advancements, such methods often require dense camera views for training and retraining for new scenes, constraining their utility in dynamic settings. GNFactor attempts to address this by introducing a voxel encoder [138], yet the challenge of dense view dependency remains. Recently, D^3Fields proposed a dynamic and semantic 3D representation through 3D fusion, aiming for real-time updates with limited viewpoints [125]. However, D^3Fields requires feature extraction at every time step, increasing computational demands and complicating high-frequency reconstruction, highlighting a critical area for improvement in dynamic scene representation for robotic manipulation.

View-Generalization for Visuomotor Policies In the field of robot learning, a primary challenge has been training models on limited views and achieving generalization to unseen views. Despite extensive efforts, such as those seen in the RoboNet [22] which amassed large-scale video datasets of various manipulation tasks, models pre-trained on these datasets still show poor performance, with success rates often below 20% on unseen camera viewpoints. Previous approaches to tackle this problem often extensive samples in simulation environments [134, 16], additional training viewpoints to create

view-agnostic representations [102, 28, 137], or requires less scalable task-related inductive bias [103, 117]. Our simpler solution to the problem is to incorporate additional depth information and construct semantic and dynamic 3D representations allowing for effective projection back to training views, thus enhancing view generalization capabilities.

5.2 Dynamic Object-centric Gaussians

5.2.1 Preliminaries on Gaussian Splatting

Our initial scene representation is constructed based on 3D Gaussian Splatting [47]. The scene is represented by a collection of 3D Gaussians, where the ith Gaussian is specified by a set of learning parameters: $\mathbf{x}_i \in \mathbb{R}^3$ is Gaussian center, $\mathbf{R}_i \in SO(3)$ the rotation, $\mathbf{s}_i \in \mathbb{R}^3$ the scale, $\mathbf{c}_i \in \mathbb{R}^3$ the color, and $\alpha_i \in \mathbb{R}$ the opacity. The weight w_i of each g_i on a point $\mathbf{p}$ in 3D space is determined by the Gaussian distribution, adjusted by the opacity:

$$w_i(\mathbf{p}) = \sigma(\alpha_i) \exp\left(-\frac{1}{2} (\mathbf{p} - \mathbf{x}_i)^\top \Sigma_i^{-1} (\mathbf{p} - \mathbf{x}_i) \right)$$

where $\sigma(\cdot)$ denotes the sigmoid function, and Σ_i is the covariance matrix, derived from its rotation and scale. To render an image I^{render} from a camera viewpoint, the 2D center of a Gaussian g_i is projected onto the image plane using the camera matrices. The 2D weight w_i^{2D} is similarly computed with the 2D center and the covariance. All the 2D centers are sorted then by depth in ascending order, and pixel color $I^{\text{render}}[u,v]$ is accumulated:

$$I^{\text{render}}[u,v] = \sum_i \mathbf{c}_i w_i^{\text{2D}}(u,v) \prod_{j=1}^{i-1} (1 - w_j^{\text{2D}}(u,v))$$

Finally, given a ground-truth image I from the viewpoint, the Gaussian parameters can be optimized by minimizing a differentiable photometric loss that measures that distance between I and I^{render}. This optimization process is fully differentiable and designed for GPU-based parallel computation, ensuring rapid training.

5.2.2 Problem Formulation and Initial Reconstruction

We seek to construct a semantic and dynamic 3D representation S_t of the scene for each time step t given views from a few RGB-D cameras. For each camera labeled with c, we have the data tuple $(I_{c,t}, D_{c,t}, E_{c,t}, K_c)$, where $I_{c,t}$ is the RGB image, $D_{c,t}$ is the depth image, $E_{c,t}$ represents the time-dependent camera extrinsic, and K_c denotes the camera intrinsic. These cameras may be static, affixed to the robot or other moving objects. Our main challenge is to update the scene at a high frequency (30 Hz).

Due to the requirement for update speed and limited camera views in robotic applications, relying solely on spatial information from the current time step is inadequate for accurate reconstruction. Our proposed solution seeks not only to reconstruct the scene S_t using spatial information but also to enrich it with temporal information from previous time steps. This is achieved by auto-regressively reconstructing S_t from S_{t-1}, thereby implicitly utilizing information from all previous time steps. By doing this, the scene representation also naturally exhibits temporal continuity, possibly allowing the agent

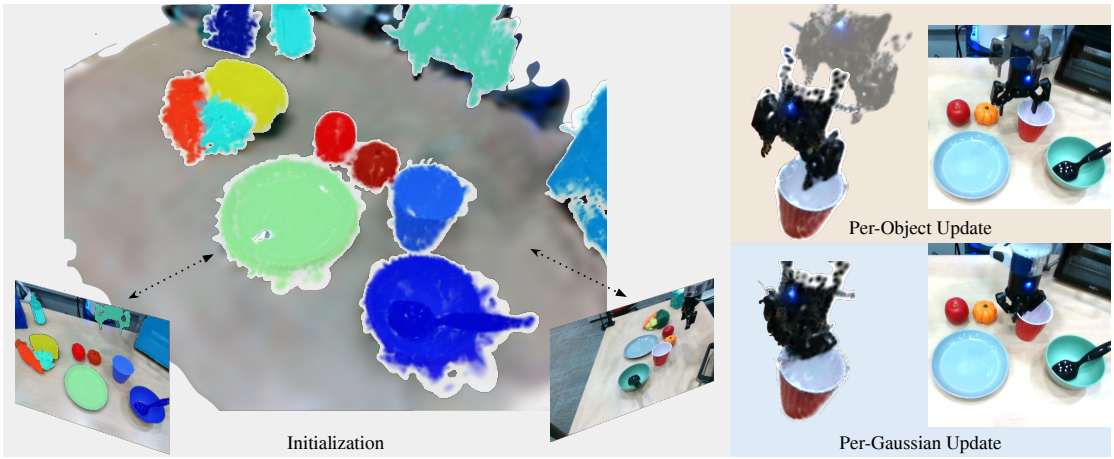


Figure 5.2: **Method Overview.** We obtain object-wise segmentation from 2D foundation models [52] at initial reconstruction. In the following updates, objects displacements are optimized with photo-metric loss. We also optimize for the displacements of individual Gaussians to account for non-rigid transformations like the closing of the robot gripper.

to capture and reflect changes over time. This also allows the computations, such as semantic extractions, at the initial time step to be carried over.

We propose to use the 3D Gaussians [47] as our scene representation: S_t is represented by a set of 3D Gaussians, $(\mathbf{x}_{i,t}, \mathbf{R}_i, \mathbf{s}_i, \mathbf{c}_i, \alpha_i)$, where the Gaussian centers are time-variant. At the initial time step, we initialize the scene with a dense point cloud obtained from the camera views. This ensures the initial reconstruction is regularized even though the views are few. We also obtain semantic features relevant to the task from 2D foundation models.

Upon obtaining the initial scene S_0, a naive approach for progressing to S_1 involves using the spatial parameters of S_0 as initial values for $\mathbf{x}_{i,1}$, and then updating these parameters with new observations $(I_{c,1}, E_{c,1}, K_c)$. This method, however, faces two primary issues: limited camera views at subsequent time steps can lead to overfitting, such as moving excess points from the background to incorrectly cover moving foreground objects; and the approach is too slow for the rapid updates required in robotics. To address these challenges, we introduce object-centric updates, as illustrated in Fig. 5.2.

Incorporating objectness into the Gaussian scene representation is a pivotal aspect of our method. Besides reconstructing the geometric scene with 3D Gaussian Splatting, the initial step in our approach also utilizes pretrained segmentation models to obtain instance segmentation of the scene. Specifically, we pick one camera view and its associated RGB image I_c, and obtain a segmentation mask M_c. The segmentation labels are then lifted into 3D space through camera matrices and depth D_c, so that each point in the point-cloud extracted, $\mathscr{P}_c$, has a corresponding segmentation label. Finally, the point clouds obtained from other views inherit their respective segmentation labels from their nearest neighbors in $\mathscr{P}_c$. Thus, each 3D Gaussian is enhanced with a segmentation label k, $g_i = (\mathbf{x}_{i,t}, \mathbf{R}_i, \mathbf{s}_i, \mathbf{c}_i, \alpha_i, l_i)$, where $l_i \in \{1, \ldots, K\}$ for K detected objects. We further label the background with $l_i = 0$. We visualize this initial segmentation on the left of Fig. 5.2, and this segmentation is carried on in the following dynamic updates, as shown in Fig. 5.3. In theory, many off-the-shelf segmenters is applicable for our purpose, but we obtain the segmentation map through GroundedSAM [52, 67, 94, 13, 81] with the language query "object". In the following sections, we introduce how to use the

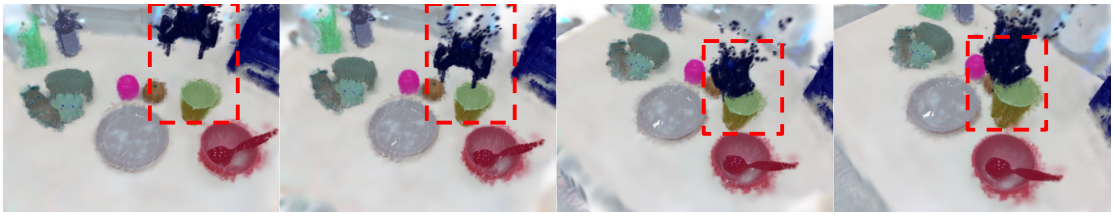


Figure 5.3: **Dynamic Segmentation.** We show the segmentation map at different time steps and rendered at different views.

Algorithm 3 Dynamic Gaussian Splatting for Real-time Robotics

Require: $n_{\text{step}} = 3$
  for time step t **do**
    Set $\delta_i := 0$ for each Gaussian i where $l_i \neq 0$
    Receive camera views $V_t = \{(I_{c,t}, E_{c,t}, K_c)\}$
    if $t = 0$ **then**
      S_0, K:= Initialize (V_t)
      Set $G_k := 0$ for each object k
    else
      for step in n_{step} **do**
        $x_{i,t} := x_{i,t-1} + G_k + \delta_i$ for $l_i = k$, for $k \in \{1, \ldots, K\}$
        Render I_c^{render} and compute loss $\mathscr{L}_c$
        Perform gradient updates: $G_k := G_k - \alpha_0 \nabla_{G_k} L_c$, $\delta_i := \delta_i - \alpha_1 \nabla_{\delta_i} \mathscr{L}_c$
      end for
    end if
    end for

segmentation information to rapidly update the scene given dynamic movements.

5.2.3 Object-centric Updates

Optimizing each individual Gaussians freely can lead to overfitting or nonphysical deformation of objects due to limited views and few number of updates. To regularize the update, we introduce G_k as the group displacement for each object k. We also introduce an individual displacement δ_i for each Gaussian g_i to account for rotations and non-rigid transforms such as the closing of the robot gripper. At a step t, G_k is initialized with the value obtained at step $t-1$ to carry over some momentum, and δ_i is initialized with zeros.

Finally, an essential modification is made for background Gaussians (labeled $l_i = 0$), which are kept fixed during optimization. This constraint is instrumental in preventing the model from overfitting by relocating background Gaussians to improperly occlude or merge with foreground objects. It ensures that the background remains stable and consistent across updates, thereby focusing the optimization process on accurately capturing and tracking the movement and deformation of objects within the scene. We summarize the pipeline in Algorithm 3. Our method achieves update rates of up to 30Hz, aligning with the dynamic needs of robotic operations.

5.3 3D-Aware Manipulation

To demonstrate the usefulness of our representation, we propose two straightforward yet effective applications of our representation to robotic manipulation. First, we show how to achieve view-robustness for image-based visuomotor policies. Second, we applies our representation to enable grasping of moving unseen objects conditioned on open-vocabulary language queries.

5.3.1 View-Robust Visuomotor Policy Learning via GSMimic

Consider a visuomotor policy which takes as inputs RGB images from a set of cameras. The problem of view-robustness arises if the training viewpoints are fixed to a coordinate frame, for example, the world frame or the end-effector frame. If the cameras are mounted differently during training time, the changes in input observation create a distribution shift that leads to significant performance drop. This issue cannot easily be handled during training without additional training cameras. With object-centric Gaussian representation, we can circumvent this issue with the additional depth input. During test-time, we can render via our 3D scene representation to get pseudo observations from the same viewpoints as training time. One of the complications is that due to limited field-of-view, test-time viewpoints will not fully cover the training viewpoints, creating empty areas in the rendering. To fix this, we directly train with renderings of foreground Gaussians only by removing Gaussians with label $l_i = 0$ during rendering. We specifically evaluate this strategy on visuomotor policies trained via behavior cloning, and term the overall approach GSMimic.

5.3.2 Language-Conditioned Dynamic Grasping

Our representation is readily applicable to zero-shot language-conditioned dynamic grasping. In this setting, a user issues a language query for the robot to grasp a specified object without prior demonstrations. The task is complicated by the possibility that the target object may be moving, requiring the agent to adapt dynamically. At the initialization stage, we extract a language-aligned feature $\mathbf{f}_k$ for each object k with CLIP [91]. Then, at query time, we use CLIP to extract an embedding $\mathbf{f}_q$ for the query, and the query is matched with the objects in the scene based on cosine distance:

$$k_q = \underset{k \in \{1,\ldots,K\}}{\arg\max} \frac{\mathbf{f}_k \cdot \mathbf{f}_q}{||\mathbf{f}_k|| \cdot ||\mathbf{q}||}$$

With the benefit of explicity 3D representation, at time step t, we are able to extract the point-cloud of the target object $\mathscr{P}_q$ by collecting the centers of Gaussians marked by $l_i = k_q$. The point-cloud forms the basis for determining a viable grasp, parameterized by a pose T_t. In particular, we randomly sample grasp poses near the point-cloud $\mathscr{P}_q$ and take the grasp with the maximal antipodal score. A motion planner is then used to direct the robot to the pose specified by T_t. Both the semantics, dynamics, and 3D aspects are crucial for the sucess of the task.

Table 5.1: **Evaluation of Simulation Tasks Given Different Testing Viewpoints.** We present success rates of tasks with 100 different initial conditions under the train view and three test views: close view (C), zoom out view (Z), zoom out and side view (S).

	Lift				Can				Square				Tool Hang			
	Train	Test Views			Train	Test Views			Train	Test Views			Train	Test Views		
		C	Z	S		C	Z	S		C	Z	S		C	Z	S
DP	**0.98**	0.47	0.0	0.0	**0.93**	0.34	0.0	0.0	**0.82**	0.23	0.0	0.0	**0.64**	0.12	0.0	0.0
DP3	0.95	0.95	0.92	0.83	0.58	0.59	0.48	0.42	0.62	0.61	0.59	0.54	0.14	0.12	0.11	0.08
GSFix	**0.98**	0.85	0.80	0.07	0.91	0.87	0.67	0.03	0.80	0.23	0.00	0.00	0.60	0.15	0.00	0.0
GSMimic	**0.98**	**0.97**	**0.94**	**0.90**	0.92	**0.94**	**0.93**	**0.85**	0.81	**0.78**	**0.77**	**0.72**	0.62	**0.60**	**0.58**	**0.52**

Table 5.2: **Evaluation of Real-World Tasks Given Different Testing Viewpoints.** We present success rates of two real-world tasks with 10 different initial conditions, similarly from the training view and 3 test views.

	Stack Cups				Unstack Cups			
	train	close	zoom out	side	train	close	zoom out	side
DP	9/10	3/10	0/10	0/10	8/10	1/10	0/10	0/10
DP3	5/10	4/10	4/10	4/10	2/10	1/10	2/10	1/10
GSMimic	8/10	9/10	8/10	6/10	8/10	8/10	7/10	5/10

5.4 Evaluation

5.4.1 View-Robust Behavior Cloning

In our experimental evaluation, we seek to investigate the generalization ability of GSMimic to unseen camera viewpoints during test time.

Simulation Evaluation. We used Robomimic [72], a large-scale robotic manipulation benchmark as our simulation testbed. We evaluated on the 4 single-arm Franka tasks from the benchmark: Lift, Can, Square, and Tool Hang. We used proficient human teleoperated demonstration dataset for each task, and use the RGB-D observation from the default "agentview" camera for the training.

Real-world Evaluation. We designed 2 tasks for real world validation on a Franka Panda Robot. (1) Cup Stacking requires the robot to pick up one of the cups on the table and place it into the other cup. (2) Cup Unstacking requires the robot to grasp the thin edge of the top cup, place it on the table, and then push it forward to roughly align with the other cup. Both tasks use Cartesian velocity control as the control space, and a proprioceptive inputs and a single front camera view as the observation space. We collect 50 tele-op demonstrations per task with a meta quest controller.

Algorithm Comparisons. We evaluated two prior methods for behavior cloning, the diffusion policy [18] as the image-based baseline, and 3D Diffusion policy (DP3) [139], which is recently proposed method that takes as inputs point-clouds. These methods demonstrate great performance in their respective input modalities. For our simulation tasks, we also evaluated an ablated version of method which we will refer to as GSFix. Instead of rendering from the foreground Gaussians, GSFix directly renders from all

of the Gaussians. For both GSFix and GSMimic, we use diffusion policy with the only difference being inputs to the model.

Evaluation Protocol. For each task, we evaluated on 4 viewpoints of increasing difficulties: train view, close view (C), zoom out view (Z), and side view (S). In each view, we ensure that the objects of interest are still in sight. Please refer to the Appendix for a visualization of the views for each task. We reported success rate of each task evaluated at 100 and 10 different starting configurations for simulation and real-world tasks, respectively.

Experimental Results

We summarized our evaluation results for simulation tasks in Table 5.1 and real-world tasks Table 5.2.

3D Understanding of the Scene is Critical for View Robustness. As seen in the results, even though diffusion policy achieves great performance given observations from the training views, the success rate drops significantly even for the close view, a small perturbation to the training view, while the policy completely fails when the views are shifting farther away. The effect is even more drastic for more high-precision tasks like Tool Hang and Cup Unstacking (which requires the gripper to grasp on a thin edge). On the other hand, GSMimic achieves comparable performance at training views, while maintaining a reasonable performance across all testing views, demonstrating the importance of our dynamic 3D representation.

Learning with 2D Inputs Improves Task Performance. Similar to GSMimic, DP3 maintains a reasonable performance across different testing viewpoint. However, the task performance is in general considerably lower than the image-based models, especially for more complicated tasks. This highlights the current gap between learning directly from RGB inputs versus 3D representations, and the gap is likely to remain due to the abundance of 2D data and models. While on the other hand, our 3D representation has the flexibility to transform into 2D inputs, thus can better leverage rich semantics and achieve better task performance.

Rendering with Foreground Only is Crucial to Avoid Distribution Shift. If we directly render the Gaussians to obtain RGB inputs for training and testing as in GSFix, the task performance is still superior compared to diffusion policy (DP) at close views. However, at harder test views, the empty areas in the rendering due to limited field-of-view cause significant distribution shift, so that GSFix similarly fails. In fact, at harder testing views like side, occlusions still cause performance drops for GSMimic. This suggests possible augmentations to further handle distribution shifts in input observation for our future works.

5.4.2 Language-conditioned Dynamic Grasping

Evaluation Setup. We evaluated our method on language-conditioned dynamic grasping on two sets of five objects from a dining and a tool scene, as shown in Fig. 5.4. We first experiment on static grasping as a baseline. Then in the dynamic setting, we randomly move around the target objects when the robot is in action. For each object and setting, we repeats for 5 trials. As a baseline comparison, we remove object-centric updates, and directly optimize for the position of each Gaussian between updates (Object-Blind).

Evaluation Results. The results is presented in Table 5.3. From the results on

Table 5.3: Evaluation of Language-conditioned Dynamic Grasping

	Dining					Tools					Total
	Green Bowl	White Bowl	Carrot	Snack	Spoon	Brush	Clamp	Screw driver	Tape	Mouse	
Static	5/5	5/5	5/5	4/5	4/5	5/5	3/5	4/5	4/5	4/5	43/50
Object-Blind	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/50
Ours	4/5	5/5	5/5	3/5	3/5	4/5	2/5	3/5	3/5	4/5	36/50

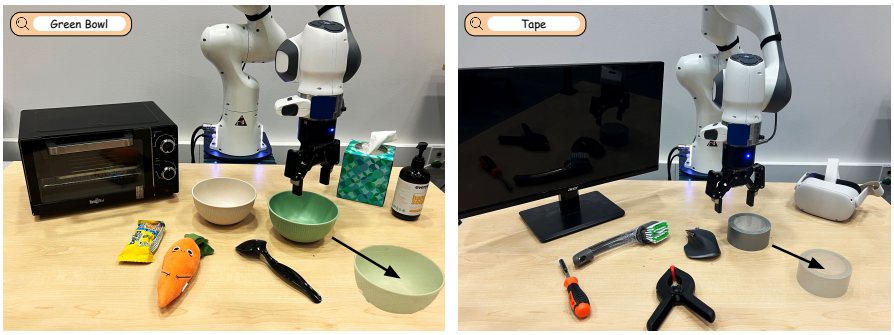


Figure 5.4: Language-conditioned Dynamic Grasping Task setup

static setting, we show that a semantic 3D representation is powerful, achieving a 86% success rate without demonstrations or other prior information. More importantly, our method still achieves a 72% success rate when objects are moving. This is only possible due to the dynamic aspect of our representation. We also show that our object-centric formulation is crucial, as the Object-Blind ablation completely fails to model object movements, making it impractical for dynamic scenes.

5.5 Discussion

In this work, we propose to leverage 3D Gaussians as a semantic and dynamic 3D representation for robotics. We achieve a high update rate of 30 Hz with object-centric initialization and updates, which is sufficient for most robotic tasks. We demonstrate the practicality of our representation for training view-robust behavior cloning policies via GSMimic and language-conditioned dynamic grasping. However, a key limitation of our method is that in its current form, it does not introduce new Gaussians to represent possible new objects, which is crucial for extending the representation to open-world manipulation. We believe that with this extension, our proposed representation has the potential to apply to a wide range of in-the-wild robotic applications.

Chapter 6

Conclusions

This thesis explored how imitation learning can serve as a flexible and scalable framework for teaching robots dexterous manipulation skills. Through a series of systems—BiDex, FACTR, DRP, and object-centric Gaussian Splatting—we tackled core challenges in demonstration collection, multimodal policy learning, closed-loop motion generation, and 3D perception. These contributions illustrate that imitation learning can lead to dexterous manipulation skills when paired with principled inductive biases and thoughtfully designed data collection pipelines.

Looking forward, one exciting direction is to extend imitation learning toward modeling the expert's underlying decision process. Human and planner experts often operate with structured reasoning—deliberating over task objectives, latent constraints, and anticipated consequences. Capturing this implicit structure may require new forms of supervision as well as learning frameworks that can internalize planning-like computation. By imitating not only what the expert does, but also how they think, robots may become more adaptive, robust, and capable of solving novel problems.

Another promising direction is to shift from learning reactive policies alone to also learning world models from offline demonstration datasets. Rather than mapping observations directly to actions, world models aim to capture the underlying dynamics and structure of the environment. Combining world models with imitation learning could lead to more general-purpose robotic systems capable of simulating outcomes, adapting to new tasks, and reasoning over long horizons.

An additional avenue for enhancing imitation learning policies is through online fine-tuning using real-world reinforcement learning (RL). While imitation learning provides a strong foundation by leveraging expert demonstrations, it can be further optimized by allowing the robot to interact with the environment and refine its policies based on active exploration. This approach enables the system to adapt to nuances and variations that were not captured in the initial training data. By integrating online RL, robots can incrementally improve their performance, learn from their mistakes, and discover more efficient strategies for task execution. This continuous learning process not only enhances the robustness and adaptability of the policies but also allows for the development of more sophisticated behaviors that are better aligned with real-world dynamics and constraints.

Appendix A

Details for BiDex

A.1 Videos, Assembly Instructions and Software on our Website

Please see our project website for videos, assembly instructions and software. This information is useful to recreate BiDex and create variants of it using high quality motion capture gloves.

A.2 Detailed Cost Analysis

Please see Table A.1 and Table A.2 for a detailed Bill of Materials and breakdown of the cost to create BiDex. This is accurate pricing as of the paper submission. While we assert that BiDex is low cost, we acknowledge that it is still not affordable for everyone such as hobbyists. We believe that the price of motion capture gloves will continue to decrease over time as technology improves and demand increases in our field as well as other adjacent fields. Altogether, the cost of the robot arms, hands, and teleop system costs around $30k and can be accessible for academic or industry labs.

A.3 User Study

To further evaluate BiDex, we conducted a user study involving novice users who tested both BiDex and the Apple Vision Pro system. Each participant performed the Pringles handover task over 10 trials, following a 3-minute practice session with each system. We collected data on average task completion times and success rates, as well as users' ratings (on a scale from 1 to 5) regarding accuracy, responsiveness, ease of use, and their confidence in each system. The results are summarized in Fig. A.4.

Our findings reveal that users completed tasks significantly faster with BiDex. While all participants achieved high success rates with BiDex, the Apple Vision Pro exhibited greater variability in performance. Notably, one user (User 5) struggled with controlling the Apple Vision Pro, resulting in a broken robot hand and a zero success rate for that trial. However, a few users managed to achieve success rates with the Apple Vision Pro that were comparable to those with BiDex. Overall, participants rated BiDex as more accurate, responsive, and easier to use, and they expressed greater confidence in using it.

Object	Quantity	Total
Pair of Manus Meta Gloves	1	$6000
Dynamixel XL330-M288 (Gello)	12	$300
U2D2 Control PCB	2	$40
5v 20A Power Supply	2	$25
14 AWG Cabling	1	$20
PLA Printer Plastic	N/A	$10
Total		$6395

Table A.1: We present the bill of materials of BiDex for two tracker arms and gloves. The total cost is around $6000, mostly due to the Manus Meta gloves.

Object	Quantity	Total
xArm 6	2	$18000
Ubuntu Laptop	1	$2000
Mobile Base	1	$6000
Zed Camera	3	$1200
LEAP Hand or LEAP Hand V2	2	$4000
Total		$31,2000

Table A.2: We present the bill of materials of the mobile robot setup. The robot and BiDex costs around $35,000 in total which we believe is reasonable for a dexterous bimanual robot hand setup with 50+ degrees of freedom.

A.4 Policy Performance Comparison

We present additional results to compare and evaluate policies trained from data collected with BiDex and the Apple Vision Pro. In particular, we collected 100 demonstrations on the Pringles can handover task using both systems, and trained policies with different numbers of demonstrations. We then evaluate each policy for 10 trials at roughly the same starting poses. We summarized our results in Fig. A.2.

In general, we found that policy performance scales with the number of demonstrations, an unsurprising result which highlights the need of efficient and effective teleoperation systems to collect large amounts of robotic data. Before conducting the evaluation, we hypothesized that policies trained on different data sources would achieve

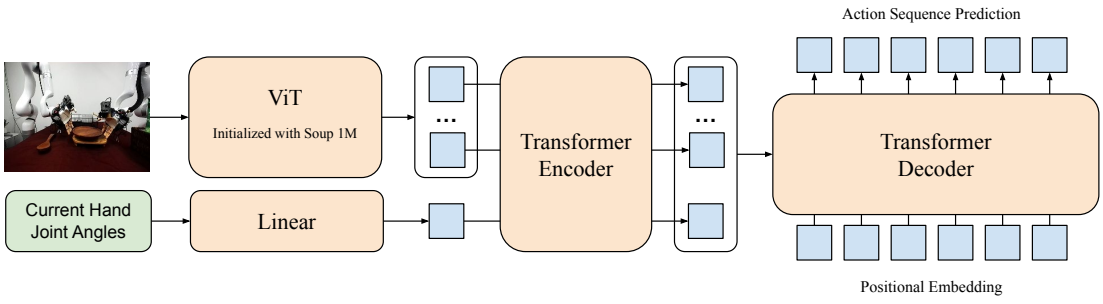


Figure A.1: Behavior cloning policy architecture.

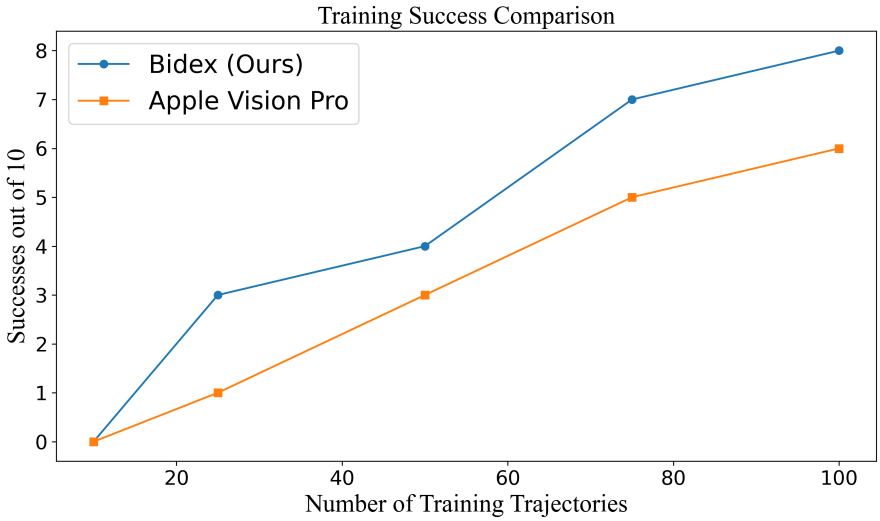


Figure A.2: Training autonomous policies with BiDex enables a higher success rate with less data than an Apple Vision Pro baseline. This is thanks to the higher quality data.

similar performance given the same number of demonstrations. However, to our surprise, policies trained from the Apple Vision Pro perform worse, mainly due to abrupt wrist movements, as shown in the "Apple Vision Pro vs Ours" section. We hypothesize that the difference in action spaces between two teleoperation systems results in the performance differences. The Apple Vision Pro commands in end-effector space, and a small prediction error can result in large errors in joint space. While BiDex commands directly in joint space, which results in smoother actions.

A.5 About Manus Glove

We use the Manus Meta Quantum Metagloves [74] which is an $6000 tracking Mocap glove. Each finger is tracked by the glove and returns the fingertip positions as xyz-quaterion and also 4 different angles for each finger $\theta_{\text{MCP}_{\text{side}}}, \theta_{\text{MCP}_{\text{fwd}}}, \theta_{\text{PIP}}, \theta_{\text{DIP}}$ using hall effect sensors with very high accuracy and at 120hz. We use their Windows API (Linux is not available at time of release) and release our version of that which sends the software to a Linux machine running Robot Operating System (ROS). These gloves are available for purchase at `https://www.manus-meta.com/` and our software is available on our project website

A.6 SteamVR Baseline

For the wrist tracking SteamVR baseline, we use the Manus Meta SteamVR trackers which connect to the gloves and seamlessly route the data through the aforementioned Windows APIs. They are wireless but require SteamVR Lighthouses setup around the perimeter of the workspace. In our test we mount the 4 SteamVR trackers on the ceiling to avoid as many occlusions as possible. We also mount the 4 trackers in a 16ft square around where the teleoperator would stand which is the recommended configuration. We will release this code for others to recreate in their comparison study.

A.7 Apple Vision Pro Baseline

The Apple Vision Pro baseline is based off of Park and Agrawal [83]. With this data, we control the hand using the same inverse kinematics as with the Manus Glove. For the arm, we scale, translate and rotate for the robot embodiment and then pass through inverse kinematics to control the arm.

A.8 Behavior Cloning Policy Architecture and Hyperparameters

We illustrate our policy architecture in Figure A.1. Our behavior cloning policy takes as input a RGB image and current hand joint angles (proprioception). We obtain tokens for the image observation via a ViT [26] and a token for joint proprioception via a linear layer. The weights of ViT is initialized from the Soup 1M model from [23]. The tokens then pass through a action chunking transformer [141], a encoder-decoder transformer, to output a sequence of actions. The action space is the absolute joint angles of two arms and two hands. A key decision that greatly improves policy generalization is to exclude current arm joints from the proprioception. Intuitively, this may force the model to extract object information from image observations, rather than overfitting to predict actions close to current arm states.

We list key hyperparameters for our behavior policy training Table A.3. In general, we are able to obtain well-performing policies with 20-50 demonstrations and 1 hour of wall-clock time training on a RTX4090. With our easy-to-use teleoperation system, we are able to obtain diverse policies for complex bimanual dexterous tasks quickly.

Hyperparameter	Value
Behavior Policy Training	
optimizer	AdamW
base learning rate	3e-4
weight decay	0.05
optimizer momentum	$\beta_1, \beta_2 = 0.9, 0.95$
batch size	64
learning rate schedule	cosine decay
total steps	10000
warmup steps	500
augmentation	GaussianBlur, Normalize, RandomResizedCrop
GPU	RTX4090 (24 gb)
Wall-clock time	$\sim$ 1 hour
Visual Backbone ViT Architecture	
patch size	16
#layers	12
#MHSA heads	12
hidden dim	768
class token	yes
positional encoding	sin cos
Action Chunking Transformer Architecture	
# encoder layers	6
# decoder layers	6
#MHSA heads	8
hidden dim	512
feedforward dim	2048
dropout	0.1
positional encoding	sin cos
action chunk	100

Table A.3: Hyperparameters for Behavior Cloning Policy Training

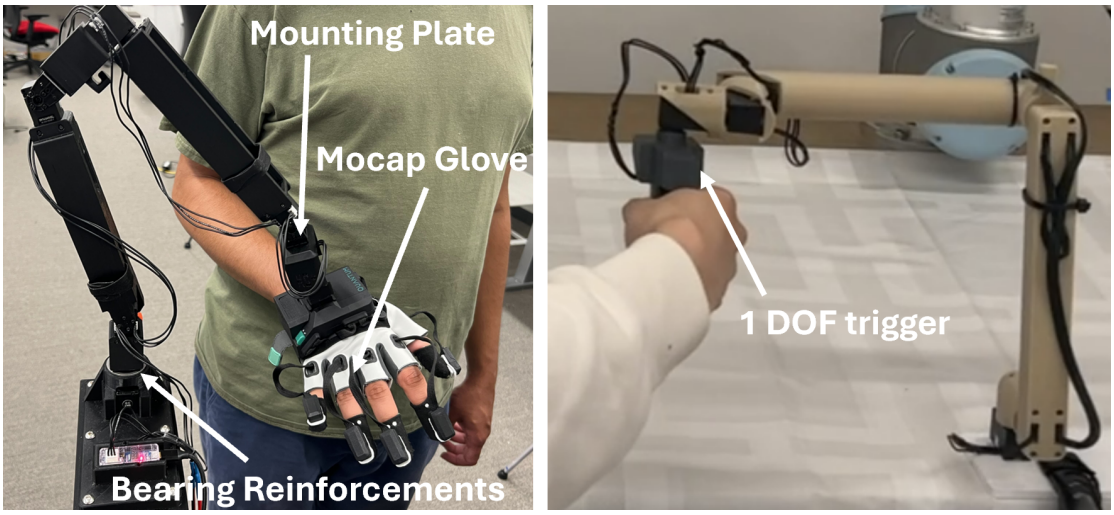
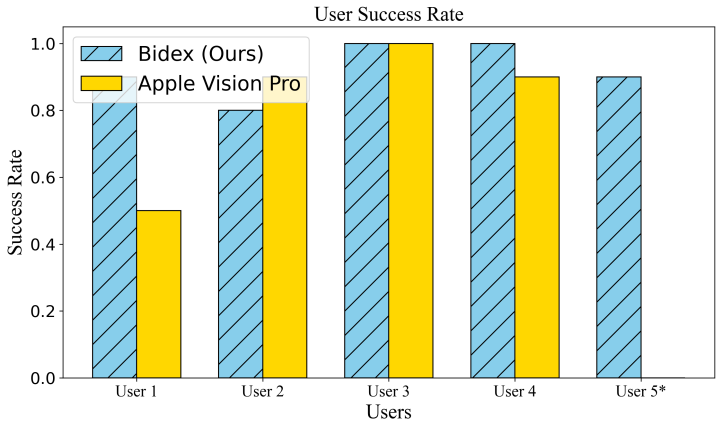
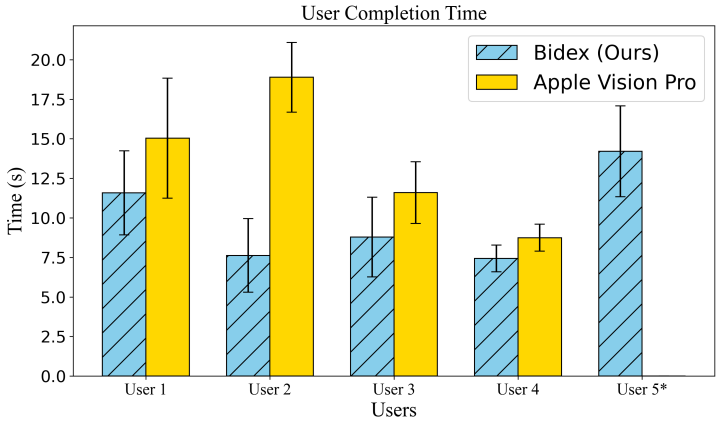


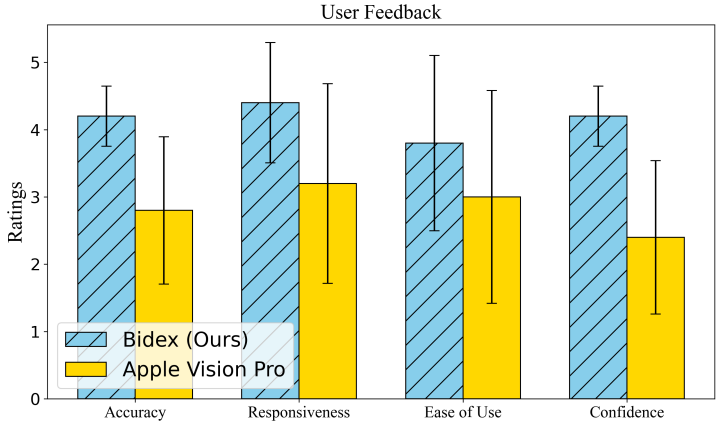
Figure A.3: We compare BiDex to the GELLO system designed for 1 DOF grippers. Our system uses a motion capture glove to capture full fingertip information and is reinforced to handle the wears and tears of this additional weight.



(a) Bidex has consistently better user success rates.



(b) Bidex is faster as seen in these user mean and standard deviation completion times.



(c) Users feedback on accuracy, responsiveness, ease of use, and confidence in using each system shows that Bidex is empirically easier to use.

Figure A.4: Novice operators were asked to complete the Pringles can handover tasks for ten trials with BiDex and the Apple Vision Pro. * User 5 found it hard to control the system with the Apple Vision Pro and broke robot hands during operation, resulting in zero success rate.

Appendix B

Details for FACTR

B.1 Analysis of FACTR from Neural Tangent Kernel (NTK) Perspective

In this section, we aim to give a sketch of a more theoretical argument for why operators like Gaussian blur or downsampling in FACTR could help the policy attending to force input. We will first briefly introduce Neural Tangent Kernel (NTK), and then use it a theoretical framework to analyze the effects of Gaussian blur as an example curriculum operator.

B.1.1 Preliminaries on Neural Tangent Kernel (NTK)

The Neural Tangent Kernel (NTK) is a theoretical framework used to analyze the behavior of neural networks, particularly in the limit of infinite width [43, 4]. For a neural network $f_\theta(x)$ with parameters θ, the NTK is defined as:

$$k(x_i, x_j) = \langle \nabla_\theta f_\theta(x_i), \nabla_\theta f_\theta(x_j) \rangle,$$

where $\nabla_\theta f_\theta(x)$ is the gradient of the network's output with respect to its parameters θ, and $\langle \cdot, \cdot \rangle$ denotes the inner product. In the infinite width limit, the NTK becomes deterministic and remains constant during training. Assuming the parameters θ are initialized from a Gaussian distribution, the NTK $k(x_i, x_j)$ converges to a deterministic kernel $k_\infty(x_i, x_j)$ given by:

$$k_\infty(x_i, x_j) = \mathbb{E}_{\theta \sim \mathcal{N}(0,I)} \left[\langle \nabla_\theta f_\theta(x_i), \nabla_\theta f_\theta(x_j) \rangle \right],$$

where the expectation is taken over the Gaussian initialization of θ.

The NTK $k(x_i, x_j)$ is a similarity function: if $k(x_i, x_j)$ is large, then the predictions $f_\theta(x_i)$ and $f_\theta(x_j)$ will tend to be close. If we have n training points (x_i, y_i), k defines a positive semi-definite (PSD) kernel matrix $K \in \mathbb{R}_+^{n \times n}$ where each entry $K_{ij} = k(x_i, x_j)$.

Fascinatingly, when we train this infinite-width neural network with gradient flow on the squared error, we precisely know the model output at any point in training. At time t, the training residual is given by:

$$f_{\theta_t}(x) - y = e^{-\eta K t}(f_{\theta_0}(x) - y),$$

where η is the learning rate, K is the kernel matrix, and $f_{\theta_0}(x)$ is the initial model output. This equation shows that the residual error decays exponentially with a rate determined by the kernel matrix K.

B.1.2 Analyzing the Effects of Gaussian Blur with NTK

We analyze the effects of Gaussian blur as an example of a curriculum operator. While we consider a simple two-layer model here, the intuition applies to more complex architectures like vision transformers (ViTs), which have a more sophisticated NTK [9].

Consider a model where the input x is first convolved with a Gaussian kernel K_σ before being passed through the network, where σ is the standard deviation of the Gaussian kernel. The output of the model is:

$$f(x) = W^T(K_\sigma * x),$$

where $K_\sigma * x$ is the convolution of x with a Gaussian kernel K_σ. Assuming that the model has infinite width and the parameters in W are initialized from a Gaussian distribution, the NTK for this model is:

$$\begin{aligned} k_\sigma(x,x') &= \mathbb{E}_{W\sim\mathcal{N}(0,I)}\left[\langle\nabla_W f(x), \nabla_W f(x')\rangle\right] \\ &= \mathbb{E}_{W\sim\mathcal{N}(0,I)}\langle K_\sigma * x, K_\sigma * x'\rangle. \end{aligned}$$

This is the dot product between the convolved inputs $K_\sigma * x$ and $K_\sigma * x'$. As σ increases, the Gaussian convolution $K_\sigma * x$ acts as a low-pass filter, attenuating high-frequency components in the input x. This causes the convolved inputs $K_\sigma * x_i$ and $K_\sigma * x_j$ to become more similar for any pair of inputs x_i and x_j. In the extreme case where $\sigma \to \infty$, $K_\sigma * x_i \approx K_\sigma * x_j$ for all x_i, x_j, and the NTK $k_\sigma(x_i, x_j)$ approaches a constant value. This implies that the kernel matrix K becomes approximately an all-ones matrix, scaled by the magnitude of the convolved inputs.

In our curriculum, we decrease σ, where the model is first exposed to smoother visual data before gradually transitioning to the original unsmoothed data. At the beginning of the curriculum, the model focuses on low-frequency patterns and robust features from visual inputs. As σ decreases to small σ, the NTK retains discriminative power, allowing the model to learn fine-grained features from visual inputs. This gradual increase in complexity can help the model learn more effectively by avoiding overfitting to the high-frequency variations in visual inputs in the early stages of training.

More rigorously, consider the limit $\sigma \to \infty$, each blurred input $K_\sigma * x_i$ converges to the same vector ϕ. Hence,

$$k_\sigma(x_i, x_j) \;=\; \langle\phi, \phi\rangle \;=\; \|\phi\|^2,$$

and the NTK matrix K becomes

$$K \;=\; \|\phi\|^2 \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix}.$$

This matrix is rank-1, with largest eigenvalue $\lambda = n\|\phi\|^2$ (for n training points) and corresponding eigenvector $v = (1, 1, \ldots, 1)$.

Recall that in the infinite-width NTK regime, the training residual r satisfies

$$r(t) \;=\; e^{-\eta K t}\, r(0),$$

where $r(0) = f_{\theta_0}(x) - y$ is the initial residual. Decompose $r(0)$ into components parallel and perpendicular to v:

$$r(0) \;=\; r_{\parallel} + r_{\perp}, \quad \text{with } r_{\parallel} \propto v \text{ and } r_{\perp} \cdot v = 0.$$

Since K is rank-1,

$$e^{-\eta K t}\, r(0) \;=\; r_{\perp} \;+\; e^{-\eta \lambda t}\, r_{\parallel}.$$

The parallel component decays exponentially at rate λ, while the perpendicular component is unchanged. This effectively learns a single global scalar for all inputs, reducing mean-squared error by matching the average label but losing discriminative power. Thus, at the early state of the curriculum, the gradient updates will focus on using the force information and updating the force encoder to maximally differentiate between inputs.

B.2 Cost Analysis of Our Teleoperation System with Force Feedback

Please see Table B.1 for a detailed Bill of Materials and breakdown of the cost to create one leader arm and leader gripper as part of our teleoperation with force feedback system. This is accurate pricing as of the paper release.

Object	Quantity	Total
Dynamixel XM430-W210-T	2	\$539.80
Dynamixel XC330-T288-T	6	\$539.40
U2D2 Control PCB	1	\$32.10
12V 20A Power Supply	1	\$24.99
FPX330-S102 Servo Bracket	1	\$8.70
Polymaker PLA PRO Filament	1	\$24.99
U2D2 Power Hub Board	1	\$19.99
14AWG Cable	1	\$23.99
3/4” Bearing	1	\$6.99
Screws	-	\$15.99
Total		\$1229.95

Table B.1: We present the bill of materials of one leader arm teleoperation device with force feedback. The total cost is around \$1229.95.

B.3 Additional Control Laws for Our Teleoperation System

Here we define the additional control laws for our teleoperation system with force feedback: friction compensation and joint limit avoidance.

B.3.1 Friction Compensation

Joint friction introduces resistive forces that impair responsiveness, making precise motion control challenging and reducing the system's intuitive feel for the operator [78]. Friction also increases the physical effort required to back-drive the motor, leading to operator fatigue during prolonged use. To this end, we use a dynamic friction model to explicitly compensate for static, Coulomb, and viscous friction [80].

We mitigate the effects of static friction by introducing a small, high-frequency oscillatory signal to the control input, which generates micro-vibrations that prevent the motors from settling into static friction states, thereby enabling smoother transitions from rest to motion [80]. The static friction compensation torque τ_{ss} is as follows:

$$\tau_{ss}^{(i)} = \begin{cases} \mu_s^{(i)} \cos(\frac{\pi t}{f}) & \text{if } \dot{q}^{(i)} < \dot{q}_s^{(i)}, \\ 0 & \text{otherwise} \end{cases} \tag{B.1}$$

where μ_s is a calibrated static friction coefficient and f is the control loop frequency, set to 500 Hz.

We also account for kinetic friction by compensating for Coulomb friction and viscous friction as follows [80]:

$$\tau_{ks}^{(i)} = \mu_c^{(i)} \text{sgn}(\dot{q}^{(i)}) + \mu_v^{(i)} \dot{q}^{(i)} \tag{B.2}$$

where μ_c is the Coulomb friction coefficient and μ_v is viscous friction coefficient.

The total friction compensation $\tau_{friction}$ is the sum of the static friction τ_{ss} and kinetic friction τ_{ks} terms.

B.3.2 Joint Limit Avoidance

We implement the following artificial potential-based control law to prevent the operator from making the leader arm go beyond the joint limits of the follower arm:

$$U(q^{(i)}) = \begin{cases} \frac{1}{2}\eta \frac{1}{(q^{(i)} - q_{\min}^{(i)})^2}, & q^{(i)} < q_{\min}^{(i)} + \Delta q \\ \frac{1}{2}\eta \frac{1}{(q_{\max}^{(i)} - q^{(i)})^2}, & q^{(i)} > q_{\max}^{(i)} - \Delta q \\ 0, & \text{otherwise} \end{cases} \tag{B.3}$$

$$\tau_{limit}^{(i)} = -\nabla_{q^{(i)}} U(q^{(i)}) \tag{B.4}$$

where $U(q^{(i)})$ is the repulsive potential function, Δq is the safety margin, and η is the scaling factor.

B.3.3 Bi-manual Follower Arms Control with Dynamic Collision Avoidance

Most existing bi-manual teleoperation systems with a leader-follower setup command the follower arms by directly setting the joint position targets to the current joint positions of the leader arm. Instead, we employ a Riemannian Motion Policy (RMP) [92] implemented in Isaac Lab [77], where the RMP dynamically generates joint-space targets for the follower arms that best match the current joint positions of the leader arms while

incorporating real-time collision avoidance. Our system prevents the follower arms from colliding with one another or with external obstacles, such as the table, regardless of the operator's actions.

B.4 Behavior Cloning Policy Architecture and Training Hyperparameters

Our behavior cloning policy takes as input a RGB image and current hand joint angles (proprioception). We obtain tokens for the image observation via a ViT [25] and a token for joint proprioception via a linear layer. The weights of ViT is initialized from the Soup 1M model from [23]. The tokens then pass through action chunking transformer, an encoder-decoder transformer, to output a sequence of actions [141]. The action space is the absolute joint angles of the two arms for box lift, the absolute angles of a single arm for non-prehensile pivot and rolling dough, and the absolute angles of a arm and the gripper for fruit pick and place. A key decision that greatly improves policy generalization is to exclude current arm joints from the proprioception. Intuitively, this may force the model to extract object information from image observations, rather than overfitting to predict actions close to current arm states.

We list key hyperparameters for our behavior policy training Table B.2. In general, we are able to obtain well-performing policies with 20000-50000 gradient steps and 2-6 hours of wall-clock time training on a RTX4090.

B.5 Additional Experiments

Adaptive Layer Norm as an Alternative to a Curriculum As an alternative to a curriculum, we experimented with Adaptive Normalization Layers [85] on the action tokens conditioned on the force input to improve the force conditioning (AdaNorm in Tab. B.3). However, we found that it creates instability in training and leads to overfitting.

Data Augmentation as alternative to curriculum. We run 10 data augmentation experiments varying noise probabilities and noise levels applied to vision input. The best of these policies, shown as DataAug in Tab. B.3. We find that policies trained with lower augmentation levels perform well on train objects but worse than FACTR on test objects, likely because low levels of augmentations fail to facilitate proper attention to force. Higher levels of augmentation policies do not perform well across train and test sets, likely because they fail to leverage vision altogether. Our curriculum approach eliminates such tradeoff between vision and force, allowing the policy to attend properly to both force and vision.

More test objects. We doubled the number of testing objects used in three of our tasks, visualized in Fig. B.1 and updated the success rate comparison figure in Tab. B.4. By conducting testing on more unseen objects and comparing with the best performing baseline (Bi-ACT), we observe that our method yields an improved generalization performance.

Figure B.1: We more than doubled the testing set size across 3 tasks.

B.6 Detailed Quantitative Results

We present the detailed evaluation results for each task in TABLE B.5, TABLE B.6, TABLE B.7, and TABLE B.8.

Hyperparameter	Value
Behavior Policy Training	
Optimizer	AdamW
Base Learning Rate	3e-4
Weight Decay	0.05
Optimizer Momentum	$\beta_1, \beta_2 = 0.9, 0.95$
Batch Size	128
Learning Rate Schedule	Cosine Decay
Total Steps	20000-50000
Warmup Steps	500
Augmentation	RandomResizeCrop
GPU	RTX4090 (24 gb)
Wall-Clock Time	2-6 hours
Visual Backbone ViT Architecture	
Patch Size	16
# Layers	12
# MHSA Heads	12
Hidden Dim	768
Class Token	Yes
Positional Encoding	sin cos
Action Chunking Transformer Architecture	
# Encoder Layers	6
# Decoder Layers	6
# MHSA Heads	8
Hidden Dim	512
Feed-Forward Dim	2048
Dropout	0.1
Positional Encoding	sin cos
Action Chunk	100

Table B.2: Policy Architecture and Training Hyperparameters

	FACTR	AdaNorm	NoiseAug
Train (%)	**90.0**	25.0	85.0
Test (%)	**77.7**	6.1	65.0

Table B.3: Pivot task training and testing performance.

	Box Lift	Pivot	Rolling Dough
ACT (Vision-Only)	35/120	30/130	0/60
Bi-ACT	68/120	76/130	41/60
FACTR (Ours)	**105/120**	**101/130**	**46/60**

Table B.4: Policy evaluation on unseen objects across 3 tasks.

	Train	Test						Train Avg	Test Avg
	Box1	Box2	Box3	Box4	Box5	Box6	Box7		
ACT (Vision-Only)	10/10	7/10	1/10	1/10	6/10	3/10	1/10	100.0%	31.7%
ACT (Vision+Force)	10/10	2/10	4/10	4/10	10/10	10/10	5/10	100.0%	58.3%
FACTR	10/10	8/10	7/10	10/10	10/10	10/10	10/10	100.0%	91.7%

Table B.5: Comparison of methods for Box Lift task.

	Train		Test					Train Avg	Test Avg
	Box1	Box2	Box3	Box4	Box5	Box6	Box7		
ACT (Vision-Only)	10/10	9/10	3/10	0/10	7/10	2/10	1/10	95.0%	26.0%
ACT (Vision+Force)	9/10	9/10	1/10	2/10	9/10	4/10	5/10	90.0%	42.0%
FACTR	9/10	9/10	6/10	5/10	10/10	7/10	10/10	90.0%	76.0%

Table B.6: Comparison of methods for Non-Prehensile Pivot task.

	Train	Test			Train Avg	Test Avg
	Obj1	Obj2	Obj3	Obj4		
ACT (Vision-Only)	5/5	0/5	4/5	0/5	100.0%	26.7%
ACT (Vision+Force)	5/5	3/5	4/5	4/5	100.0%	73.3%
FACTR	5/5	4/5	5/5	5/5	100.0%	93.3%

Table B.7: Comparison of methods for Fruit Pick-Place task.

	Train		Test		Train Avg	Test Avg
	Obj1	Obj2	Obj3	Obj4		
ACT (Vision-Only)	0/5	0/5	0/5	0/5	0.0%	0.0%
ACT (Vision+Force)	4/5	4/5	3/5	4/5	80.0%	70.0%
FACTR	5/5	4/5	4/5	4/5	90.0%	80.0%

Table B.8: Comparison of methods for Rolling Dough task.

Appendix C

Details for Deep Reactive Policy

C.1 Detailed Task Descriptions

We propose **DRPBench** in simulation which comprises of five tasks:

- **Static Environments (SE):** These scenarios feature challenging fixed obstacles, evaluating policies performance in predictable, unchanging settings.
- **Suddenly Appearing Obstacle (SAO):** Obstacles appear suddenly ahead of the robot, directly blocking its path and requiring dynamic trajectory adaptation. This tests the policy's ability to react to unexpected changes in the environment.
- **Floating Dynamic Obstacles (FDO):** Obstacles move randomly throughout the environment, challenging the robot's reactive capabilities and its ability to avoid collisions in real time.
- **Goal Blocking (GB):** The goal is temporarily obstructed by an obstacle, and the robot must approach as closely as possible without colliding.
- **Dynamic Goal Blocking (DGB):** After reaching the goal, the robot encounters a moving obstacle and must avoid it before safely returning to the goal, testing its ability to remain reactive even after task completion.

C.2 Baseline Comparison Details

DRP is compared against a broad set of baselines, including both classical planning methods and learning-based approaches. For *classical methods*, we consider the following:

- **AIT*** [115]: A state-of-the-art sampling-based planner that requires pre-computation and precise object models of the obstacles. We enforce an 80-second limit for this pre-computation.
- **cuRobo** [116]: A state-of-the-art optimization-based planner, also used as the expert policy for generating our pretraining dataset.
- **cuRobo-Vox** [116]: Instead of precise object models of the obstacles, we used voxelized inputs derived from point clouds to enable deployment in unstructured environments.

- **RMP** [92]: A state-of-the-art locally-reactive method.

For *learning-based methods*, we compare against several recent works:

- **MπNets** [29]: A state-of-the-art neural policy, though trained on a less diverse dataset.
- **MπFormer** [30]: A follow-up to MπNets that adopts a more modern architecture and incorporates a teacher-student fine-tuning stage with hard negative mining. It trains separate policies for different task scenarios; we report the best-performing checkpoint for each task.
- **NeuralMP** [21]: An end-to-end motion generation policy trained on a more diverse dataset, but requiring a pre-computation phase known as test-time optimization (TTO) to boost performance.

C.3 DCP-RMP Details

A Riemannian Motion Policy (RMP) is a non-learning-based motion policy that takes the robot and environment state as input and outputs joint-space accelerations. RMP naturally fuses multiple single-purpose motion policies—such as moving toward a goal or repelling from an obstacle—into a single combined policy.

Each single-purpose policy is paired with a state-dependent priority metric. These policies are first transformed into a common space (typically joint space), weighted by their respective metrics, and summed together. The result is effectively a metric-weighted combination that allows selective prioritization or suppression of single-purpose policies based on the current state.

In DRP-RMP, we combine two hand-designed single-purpose policies: one for goal attraction and one for dynamic obstacle avoidance.

Goal-Attracting Policy. We define a simple goal-attracting acceleration policy $\mathbf{f}_g$ in joint space $\mathscr{Q}$, which pulls the robot joint position towards a desired joint position. It has an associated metric $\mathbf{M}_g$:

$$\ddot{\mathbf{q}}_g = \mathbf{f}_g(\mathbf{q}, \dot{\mathbf{q}}) = k_g(\mathbf{q}_g - \mathbf{q}) - k_d\dot{\mathbf{q}} \quad \text{and} \quad \mathbf{M}_g = \mu_g\mathbf{I}$$

Dynamic Obstacle-Avoiding Policy. Traditional implementations of obstacle-avoidance policy requires ground-truth obstacle information such as obstacle models and poses. In order to deploy this policy in the real-world, we propose a point-cloud based policy. Specifically, we define the task-space of this policy $\mathscr{R}$ and its associated task-space Jacobian $\mathbf{J}_r$ with respect to the joint space $\mathscr{Q}$ as follows:

$$\mathbf{x}_r = ||\mathbf{x}_p - \mathbf{x}_{obs}||^2 \quad \text{and} \quad \mathbf{J}_r(\mathbf{q}) = \frac{\partial \mathbf{x}_r}{\partial \mathbf{q}} = 2(\mathbf{x}_p - \mathbf{x}_{obs})^\top \mathbf{J}_p(\mathbf{q})$$

where $\mathbf{x}_{obs}$ is the closest dynamic obstacle point to the robot and $\mathbf{x}_p$ is the closest point on the robot surface to $\mathbf{x}_{obs}$. To extract $\mathbf{x}_{obs}$, we build a KD-Tree from the scene point cloud captured in the previous frame and query it using the current frame's point cloud to identify points not present previously, thereby classifying them as dynamic points. From these dynamic points, we select the one closest to the robot surface. We define a dynamic obstacle-avoiding acceleration policy $\mathbf{f}_r$ and its metric as $\mathbf{M_r}$ as follows:

	SE	SAO	FDO	GB	DGB
NeuralMP [21]	75.56	76.00	57.50	37.73	57.00
IMPACT	**84.60**	**86.00**	**75.50**	**66.67**	**65.25**

Table C.1: Architecture: IMPACT vs. NeuralMP.

$$\ddot{\mathbf{x}}_r = \mathbf{f}_r(\mathbf{x}_r, \dot{\mathbf{x}}_r) = k_p \exp(-\mathbf{x}_r/\ell_p) - k_v \left[1 - \frac{1}{1+\exp(-\dot{\mathbf{x}}_r/l_v)}\right] \frac{\dot{\mathbf{x}}_r}{\mathbf{x}_r/\ell_d + \varepsilon_d}$$

$$\mathbf{M}_\mathbf{r}(\mathbf{x}, \dot{\mathbf{x}}_r) = \left[1 - \frac{1}{1+\exp(-\dot{\mathbf{x}}_r/l_v)}\right] g(\mathbf{x}_r) \frac{\mu_r}{\mathbf{x}_r/\ell_m + \varepsilon_m}, \quad g(x) = \begin{cases} \frac{(\mathbf{x}_r - r)^2}{r^2}, & x \leq r \\ 0, & x > r \end{cases}$$

In essence, the obstacle-avoidance policy $\mathbf{f}_r(\mathbf{x}_r, \dot{\mathbf{x}}r)$ generates accelerations pushing away from $\mathbf{x}$obs that grow stronger as the distance $\mathbf{x}_r$ decreases. Additionally, the repulsive acceleration increases if $\mathbf{x}_r$ is closing more rapidly. The associated metric $\mathbf{M}_r(\mathbf{x}_r, \dot{\mathbf{x}}_r)$ amplifies the influence of this policy on the combined RMP when the closing speed increases and fully deactivates the policy when the obstacle is beyond a threshold distance r.

Since the dynamic obstacle-avoiding policy is defined in task-space $\mathscr{R}$, we need to transform it back to joint-space $\mathscr{Q}$ to be combined with $\mathbf{f}_g(\mathbf{q}, \dot{\mathbf{q}})$. To do this, we use the pull-back operator to transform both $\mathbf{f}_r(\mathbf{x}_r, \dot{\mathbf{x}}_r)$ and $\mathbf{M}_\mathbf{r}(\mathbf{x}, \dot{\mathbf{x}})_r$ into $\mathscr{Q}$ as follows:

$$^{\mathscr{Q}}\mathbf{f}_r = \text{pull}_{\mathscr{Q}}(\mathbf{f}_r(\mathbf{x}_r, \dot{\mathbf{x}}_r)) = (\mathbf{J}_r^\top \mathbf{M}_r \mathbf{J}_r)^\dagger \mathbf{J}_r^\top \mathbf{M}_r \mathbf{f}_r$$

$$^{\mathscr{Q}}\mathbf{M}_r = \text{pull}_{\mathscr{Q}}(\mathbf{M}_r(\mathbf{x}_r, \dot{\mathbf{x}}_r)) = \mathbf{J}_r^\top \mathbf{M}_r \mathbf{J}_r$$

Combining Policies. Given the goal-attracting policy $\mathbf{f}_g$ and the dynamic obstacle-avoiding policy transformed in joint-space $^{\mathscr{Q}}\mathbf{f}_r$, we can combine these two policies to yield the full DCP-RMP as follows:

$$\ddot{\mathbf{q}}_{mg}(t) = \left(^{\mathscr{Q}}\mathbf{M}_r + \mathbf{M}_g\right)^\dagger \left(^{\mathscr{Q}}\mathbf{M}_r^{\mathscr{Q}}\mathbf{f}_r + \mathbf{M}_g\mathbf{f}_g\right)$$

Finally, we perform Euler-integration to yield the modified joint goal $\mathbf{q}_{md}$. We note that in practice, we perform multiple Euler-integration steps per control-loop iteration to result in a more reactive behavior. In essence, $\mathbf{q}_{mg}$ prioritizes reactive dynamic obstacle avoidance if obstacles move close to the robot; otherwise, $\mathbf{q}_{mg}$ prioritizes reaching towards the desired joint position goal $\mathbf{q}_g$.

$$\mathbf{q}_{mg}(t+1) = \text{EulerIntegrate}(\mathbf{q}_{mg}(t), \dot{\mathbf{q}}_{mg}(t), \ddot{\mathbf{q}}_{mg}(t))$$

C.4 Architecture Ablation

We compare IMPACT to the LSTM-GMM architecture used in the state-of-the-art neural motion policy NeuralMP [21], holding all other factors— from dataset to pretraining and finetuning—constant to isolate the effect of architecture. As shown in Table C.1, IMPACT consistently outperforms NeuralMP across all evaluation tasks, showcasing the advantages of its architectural design.

Bibliography

[1] Ananye Agarwal, Shagun Uppal, Kenneth Shaw, and Deepak Pathak. Dexterous functional grasping. In *Conference on Robot Learning*, pages 3453–3467. PMLR, 2023.

[2] AgileX Robotics. Ranger mini. `https://global.agilex.ai/products/ranger-mini`, 2023. Omnidirectional mobile robot platform.

[3] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[4] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *Advances in neural information processing systems*, 32, 2019.

[5] Sridhar Pandian Arunachalam, Irmak Güzey, Soumith Chintala, and Lerrel Pinto. Holo-dex: Teaching dexterity with immersive mixed reality. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5962–5969. IEEE, 2023.

[6] Mohak Bhardwaj, Balakumar Sundaralingam, Arsalan Mousavian, Nathan D Ratliff, Dieter Fox, Fabio Ramos, and Byron Boots. Storm: An integrated framework for fast joint-space model-predictive control for reactive manipulation. In *Conference on Robot Learning*, pages 750–759. PMLR, 2022.

[7] Robert Bohlin and Lydia E Kavraki. Path planning using lazy prm. In *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*, volume 1, pages 521–528. IEEE, 2000.

[8] Jonathan Bohren, Radu Bogdan Rusu, E Gil Jones, Eitan Marder-Eppstein, Caroline Pantofaru, Melonee Wise, Lorenz Mösenlechner, Wim Meeussen, and Stefan Holzer. Towards autonomous robotic butlers: Lessons learned with the pr2. In *2011 IEEE International Conference on Robotics and Automation*, pages 5568–5575. IEEE, 2011.

[9] Enric Boix-Adsera, Omid Saremi, Emmanuel Abbe, Samy Bengio, Etai Littwin, and Joshua Susskind. When can transformers reason with abstract symbols? *International Conference on Learning Representations*, 2023.

[10] Gerald Brantner and Oussama Khatib. Controlling ocean one: Human–robot collaboration for deep-sea manipulation. *Journal of Field Robotics*, 38(1):28–51, 2021.

[11] Samuel R Buss and Jin-Su Kim. Selectively damped least squares for inverse kinematics. *Journal of Graphics tools*, 10(3):37–49, 2005.

[12] Berk Calli, Arjun Singh, James Bruce, Aaron Walsman, Kurt Konolige, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Yale-cmu-berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research*, 36(3):261–268, 2017.

[13] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9650–9660, 2021.

[14] Joao Carvalho, An T Le, Mark Baierl, Dorothea Koert, and Jan Peters. Motion planning diffusion: Learning and planning of robot motions with diffusion models. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1916–1923. IEEE, 2023.

[15] Constantinos Chamzas, Zachary Kingston, Carlos Quintero-Peña, Anshumali Shrivastava, and Lydia E Kavraki. Learning sampling distributions using local 3d workspace decompositions for motion planning in high dimensions. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1283–1289. IEEE, 2021.

[16] Boyuan Chen, Pieter Abbeel, and Deepak Pathak. Unsupervised learning of visual 3d keypoints for control. In *International Conference on Machine Learning*, pages 1539–1549. PMLR, 2021.

[17] Claire Chen, Zhongchun Yu, Hojung Choi, Mark Cutkosky, and Jeannette Bohg. Dexforce: Extracting force-informed actions from kinesthetic demonstrations for dexterous manipulation. *arXiv preprint arXiv:2501.10356*, 2025.

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

[19] Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *arXiv preprint arXiv:2402.10329*, 2024.

[20] Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *Robotics: Science and Systems*, 2024.

[21] Murtaza Dalal, Jiahui Yang, Russell Mendonca, Youssef Khaky, Ruslan Salakhutdinov, and Deepak Pathak. Neural mp: A generalist neural motion planner. *arXiv preprint arXiv:2409.05864*, 2024.

[22] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. In *Conference on Robot Learning*, pages 885–897. PMLR, 2020.

[23] Sudeep Dasari, Mohan Kumar Srirama, Unnat Jain, and Abhinav Gupta. An unbiased look at datasets for visuo-motor pre-training. In *Conference on Robot Learning*, pages 1183–1198. PMLR, 2023.

[24] Runyu Ding, Yuzhe Qin, Jiyue Zhu, Chengzhe Jia, Shiqi Yang, Ruihan Yang, Xiaojuan Qi, and Xiaolong Wang. Bunny-visionpro: Real-time bimanual dexterous teleoperation for imitation learning. *arXiv preprint arXiv:2407.03162*, 2024.

[25] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

[26] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[27] Anca D Dragan, Nathan D Ratliff, and Siddhartha S Srinivasa. Manipulation planning with goal sets using constrained trajectory optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 4582–4588. IEEE, 2011.

[28] Danny Driess, Ingmar Schubert, Pete Florence, Yunzhu Li, and Marc Toussaint. Reinforcement learning with neural radiance fields. *Advances in Neural Information Processing Systems*, 35:16931–16945, 2022.

[29] Adam Fishman, Adithyavairavan Murali, Clemens Eppner, Bryan Peele, Byron Boots, and Dieter Fox. Motion policy networks. In *conference on Robot Learning*, pages 967–977. PMLR, 2023.

[30] Adam Fishman, Aaron Walsman, Mohak Bhardwaj, Wentao Yuan, Balakumar Sundaralingam, Byron Boots, and Dieter Fox. Avoid everything: Model-free collision avoidance with expert-guided fine-tuning. In *CoRL Workshop on Safe and Robust Robot Learning for Operation in the Real World*, 2024.

[31] Cinzia Freschi, Vincenzo Ferrari, Franca Melfi, Mauro Ferrari, Franco Mosca, and Alfred Cuschieri. Technical review of the da vinci surgical telemanipulator. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 9(4):396–406, 2013.

[32] Zipeng Fu, Tony Z Zhao, and Chelsea Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *Conference on Robot Learning*, 2024.

[33] Jonathan D. Gammell, Siddhartha S. Srinivasa, and Timothy D. Barfoot. Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3067–3074, 2015.

[34] Ankur Handa, Karl Van Wyk, Wei Yang, Jacky Liang, Yu-Wei Chao, Qian Wan, Stan Birchfield, Nathan Ratliff, and Dieter Fox. Dexpilot: Vision-based teleoperation of dexterous robotic hand-arm system. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9164–9170. IEEE, 2020.

[35] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[36] Zihao He, Hongjie Fang, Jingjing Chen, Hao-Shu Fang, and Cewu Lu. Foar: Force-aware reactive policy for contact-rich robotic manipulation. *arXiv preprint arXiv:2411.15753*, 2024.

[37] Neville Hogan. Impedance control: An approach to manipulation. In *1984 American control conference*, pages 304–313. IEEE, 1984.

[38] Yifan Hou, Zeyi Liu, Cheng Chi, Eric Cousineau, Naveen Kuppuswamy, Siyuan Feng, Benjamin Burchfiel, and Shuran Song. Adaptive compliance policy: Learning approximate compliance for diffusion guided control. *arXiv preprint arXiv:2410.09309*, 2024.

[39] Huang Huang, Balakumar Sundaralingam, Arsalan Mousavian, Adithyavairavan Murali, Ken Goldberg, and Dieter Fox. Diffusionseeder: Seeding motion optimization with diffusion for rapid motion planning. *arXiv preprint arXiv:2410.16727*, 2024.

[40] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

[41] J. Ichnowski, Y. Avigal, J. Kerr, and K. Goldberg. Dexnerf: Using a neural radiance field to grasp transparent objects. In *5th Annual Conference on Robot Learning*, 2021.

[42] Brian Ichter, Pierre Sermanet, and Corey Lynch. Broadly-exploring, local-policy trees for long-horizon task planning. *arXiv preprint arXiv:2010.06491*, 2020.

[43] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[44] Jacob J Johnson, Linjun Li, Fei Liu, Ahmed H Qureshi, and Michael C Yip. Dynamically constrained motion planning networks for non-holonomic robots. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6937–6943. IEEE, 2020.

[45] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on robot learning*, pages 651–673. PMLR, 2018.

[46] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

[47] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (TOG)*, 42(4):1–14, 2023.

[48] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik. Lerf: Language embedded radiance fields. In *International Conference on Computer Vision (ICCV)*, 2023.

[49] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings. 1985 IEEE international conference on robotics and automation*, volume 2, pages 500–505. IEEE, 1985.

[50] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987.

[51] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[52] 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. *arXiv:2304.02643*, 2023.

[53] Masato Kobayashi, Thanpimon Buamanee, and Takumi Kobayashi. Alpha-α and bi-act are all you need: Importance of position and force information control for imitation learning of unimanual and bimanual robotic manipulation with low-cost system. *arXiv preprint arXiv:2411.09942*, 2024.

[54] Sven Koenig and Maxim Likhachev. A new principle for incremental heuristic search: Theoretical results. In *ICAPS*, pages 402–405, 2006.

[55] James J Kuffner and Steven M LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 995–1001. IEEE, 2000.

[56] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

[57] Rahul Kumar, Aditya Mandalika, Sanjiban Choudhury, and Siddhartha Srinivasa. Lego: Leveraging experience in roadmap generation for sampling-based planning. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1488–1495. IEEE, 2019.

[58] Steven LaValle. Rapidly-exploring random trees: A new tool for path planning. *Research Report 9811*, 1998.

[59] Steven M LaValle and James J Kuffner. Rapidly-exploring random trees: Progress and prospects: Steven m. lavalle, iowa state university, a james j. kuffner, jr., university of tokyo, tokyo, japan. *Algorithmic and computational robotics*, pages 303–307, 2001.

[60] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[61] Kelin Li, Shubham M Wagh, Nitish Sharma, Saksham Bhadani, Wei Chen, Chang Liu, and Petar Kormushev. Haptic-act: Bridging human intuition with compliant robotic manipulation via immersive vr. *arXiv preprint arXiv:2409.11925*, 2024.

[62] Jacky Liang, Jeffrey Mahler, Michael Laskey, Pusong Li, and Ken Goldberg. Using dvrk teleoperation to facilitate deep learning of automation tasks for an industrial robot. In *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, pages 1–8, 2017.

[63] Maxim Likhachev, David I Ferguson, Geoffrey J Gordon, Anthony Stentz, and Sebastian Thrun. Anytime dynamic a*: An anytime, replanning algorithm. In *ICAPS*, volume 5, pages 262–271, 2005.

[64] Maxim Likhachev, Geoffrey J Gordon, and Sebastian Thrun. Ara*: Anytime a* with provable bounds on sub-optimality. *Advances in neural information processing systems*, 16, 2003.

[65] Toru Lin, Yu Zhang, Qiyang Li, Haozhi Qi, Brent Yi, Sergey Levine, and Jitendra Malik. Learning visuotactile skills with two multifingered hands. *arXiv:2404.16823*, 2024.

[66] Huihan Liu, Soroush Nasiriany, Lance Zhang, Zhiyao Bao, and Yuke Zhu. Robot learning on the job: Human-in-the-loop autonomy and learning during deployment. *arXiv preprint arXiv:2211.08416*, 2022.

[67] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023.

[68] Wenhai Liu, Junbo Wang, Yiming Wang, Weiming Wang, and Cewu Lu. Forcemimic: Force-centric imitation learning with force-motion capture system for contact-rich manipulation. *arXiv preprint arXiv:2410.07554*, 2024.

[69] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *arXiv preprint arXiv:2308.09713*, 2023.

[70] Kevin M. Lynch and Frank C. Park. *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, USA, 1st edition, 2017.

[71] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[72] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning*, pages 1678–1690. PMLR, 2022.

[73] Pragna Mannam, Kenneth Shaw, Dominik Bauer, Jean Oh, Deepak Pathak, and Nancy Pollard. Designing anthropomorphic soft hands through interaction. In *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*, pages 1–8, 2023.

[74] Manus. `https://www.manus-meta.com/`. [VR haptic gloves].

[75] Matthew T Mason. Compliance and force control for computer controlled manipulators. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(6):418–432, 1981.

[76] 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 *European Conference on Computer Vision (ECCV)*, 2020.

[77] 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.

[78] G. Morel and S. Dubowsky. The precise control of manipulators with joint friction: a base force/torque sensor method. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, pages 360–365 vol.1, 1996.

[79] Movella. `https://www.movella.com/products/xsens#overview`. [Motion capture technology].

[80] H. Olsson, K.J. Åström, C. Canudas de Wit, M. Gäfvert, and P. Lischinsky. Friction models and friction compensation. *European Journal of Control*, 4(3):176–195, 1998.

[81] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[82] Lorenzo Pagliara, Enrico Ferrentino, Andrea Chiacchio, and Giovanni Russo. Safe haptic teleoperations of admittance controlled robots with virtualization of the force feedback. *arXiv preprint arXiv:2404.07672*, 2024.

[83] Younghyo Park and Pulkit Agrawal. Using apple vision pro to train and control robots, 2024.

[84] Georgios Pavlakos, Dandan Shan, Ilija Radosavovic, Angjoo Kanazawa, David Fouhey, and Jitendra Malik. Reconstructing hands in 3D with transformers. In *CVPR*, 2024.

[85] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[86] Psyonic. `https://www.psyonic.io`. [Bionic hand].

[87] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[88] Yuzhe Qin, Wei Yang, Binghao Huang, Karl Van Wyk, Hao Su, Xiaolong Wang, Yu-Wei Chao, and Dietor Fox. Anyteleop: A general vision-based dexterous robot arm-hand teleoperation system. *arXiv preprint arXiv:2307.04577*, 2023.

[89] Ahmed H Qureshi, Jiangeng Dong, Austin Choe, and Michael C Yip. Neural manipulation planning on constraint manifolds. *IEEE Robotics and Automation Letters*, 5(4):6089–6096, 2020.

[90] Ahmed H Qureshi, Anthony Simeonov, Mayur J Bency, and Michael C Yip. Motion planning networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2118–2124. IEEE, 2019.

[91] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision. *Proceedings of the 38th International Conference on Machine Learning*, 139:8748–8763, 2021.

[92] Nathan D Ratliff, Jan Issac, Daniel Kappler, Stan Birchfield, and Dieter Fox. Riemannian motion policies. *arXiv preprint arXiv:1801.02854*, 2018.

[93] Harish Ravichandar, Athanasios S Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. *Annual review of control, robotics, and autonomous systems*, 3:297–330, 2020.

[94] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, Zhaoyang Zeng, Hao Zhang, Feng Li, Jie Yang, Hongyang Li, Qing Jiang, and Lei Zhang. Grounded sam: Assembling open-world models for diverse visual tasks, 2024.

[95] Rokoko. `https://www.rokoko.com/`. [Motion capture hardware and software].

[96] Javier Romero, Dimitrios Tzionas, and Michael J Black. Embodied hands: Modeling and capturing hands and bodies together. *arXiv preprint arXiv:2201.02610*, 2022.

[97] Yu Rong, Takaaki Shiratori, and Hanbyul Joo. Frankmocap: A monocular 3d whole-body pose estimation system via regression and integration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1749–1759, 2021.

[98] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[99] Kallol Saha, Vishal Mandadi, Jayaram Reddy, Ajit Srikanth, Aditya Agarwal, Bipasha Sen, Arun Singh, and Madhava Krishna. Edmp: Ensemble-of-costs-guided diffusion for motion planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10351–10358. IEEE, 2024.

[100] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014.

[101] Shadow Robot Company. `https://www.shadowrobot.com/`. [Robotic hand].

[102] Jinghuan Shang and Michael S Ryoo. Self-supervised disentangled representation learning for third-person imitation learning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 214–221. IEEE, 2021.

[103] Pratyusha Sharma, Deepak Pathak, and Abhinav Gupta. Third-person visual imitation learning via decoupled hierarchical controller. *Advances in Neural Information Processing Systems*, 32, 2019.

[104] S. Sharma, A. Rashid, C. M. Kim, J. Kerr, L. Y. Chen, A. Kanazawa, and K. Goldberg. Language embedded radiance fields for zero-shot task-oriented grasping. In *7th Annual Conference on Robot Learning*, 2023.

[105] Kenneth Shaw, Ananye Agarwal, and Deepak Pathak. Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning. *arXiv preprint arXiv:2309.06440*, 2023.

[106] Kenneth Shaw, Shikhar Bahl, Aravind Sivakumar, Aditya Kannan, and Deepak Pathak. Learning dexterity from human hand motion in internet videos. *The International Journal of Robotics Research*, 43(4):513–532, 2024.

[107] 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.

[108] Kenneth Shaw and Deepak Pathak. LEAP hand v2: Dexterous, low-cost anthropomorphic hybrid rigid soft hand for robot learning. In *2nd Workshop on Dexterous Manipulation: Design, Perception and Control (RSS)*, 2024.

[109] W. Shen, G. Yang, A. Yu, J. Wong, L. P. Kaelbling, and P. Isola. Distilled feature fields enable few-shot manipulation. In *7th Annual Conference on Robot Learning*, 2023.

[110] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023.

[111] Bruno Siciliano, Oussama Khatib, and Torsten Kröger. *Springer handbook of robotics*, volume 200. Springer, 2008.

[112] A. Simeonov, Y. Du, A. Tagliasacchi, J. B. Tenenbaum, A. Rodriguez, P. Agrawal, and V. Sitzmann. Neural descriptor fields: Se(3)-equivariant object representations for manipulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6394–6400. IEEE, 2022.

[113] Aravind Sivakumar, Kenneth Shaw, and Deepak Pathak. Robotic telekinesis: Learning a robotic hand imitator by watching humans on youtube. In *Robotics: Science and Systems*, 2022.

[114] Shuran Song, Andy Zeng, Johnny Lee, and Thomas Funkhouser. Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations. *IEEE Robotics and Automation Letters*, 5(3):4978–4985, 2020.

[115] Marlin P Strub and Jonathan D Gammell. Adaptively informed trees (ait*): Fast asymptotically optimal path planning through adaptive heuristics. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3191–3198. IEEE, 2020.

[116] Balakumar Sundaralingam, Siva Kumar Sastry Hari, Adam Fishman, Caelan Garrett, Karl Van Wyk, Valts Blukis, Alexander Millane, Helen Oleynikova, Ankur Handa, Fabio Ramos, et al. Curobo: Parallelized collision-free robot motion generation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8112–8119. IEEE, 2023.

[117] Hsiao-Yu Fish Tung, Zhou Xian, Mihir Prabhudesai, Shamit Lal, and Katerina Fragkiadaki. 3d-oes: Viewpoint-invariant object-factorized environment simulators. *arXiv preprint arXiv:2011.06464*, 2020.

[118] Valve Corporation. `https://store.steampowered.com/steamvr`. [Virtual reality platform].

[119] Karl Van Wyk, Mandy Xie, Anqi Li, Muhammad Asif Rana, Buck Babich, Bryan Peele, Qian Wan, Iretiayo Akinola, Balakumar Sundaralingam, Dieter Fox, et al. Geometric fabrics: Generalizing classical mechanics to capture the physics of behavior. *IEEE Robotics and Automation Letters*, 7(2):3202–3209, 2022.

[120] Vicon. `https://www.vicon.com/`. [Motion capture technology].

[121] Quan Vuong, Sergey Levine, Homer Rich Walke, Karl Pertsch, Anikait Singh, Ria Doshi, Charles Xu, Jianlan Luo, Liam Tan, Dhruv Shah, et al. Open x-embodiment: Robotic learning datasets and rt-x models. In *Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition@ CoRL2023*, 2023.

[122] Homer Rich Walke, Kevin Black, Tony Z Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, Andre Wang He, Vivek Myers, Moo Jin Kim, Max Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pages 1723–1736. PMLR, 2023.

[123] Chen Wang, Haochen Shi, Weizhuo Wang, Ruohan Zhang, Li Fei-Fei, and C Karen Liu. Dexcap: Scalable and portable mocap data collection system for dexterous manipulation. *Robotics: Science and Systems*, 2024.

[124] Weiyao Wang, Du Tran, and Matt Feiszli. What makes training multi-modal classification networks hard? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12695–12705, 2020.

[125] Yixuan Wang, Zhuoran Li, Mingtong Zhang, Katherine Driggs-Campbell, Jiajun Wu, Li Fei-Fei, and Yunzhu Li. D3 fields: Dynamic 3d descriptor fields for zero-shot generalizable robotic manipulation. *arXiv preprint arXiv:2309.16118*, 2023.

[126] Y. Wi, P. Florence, A. Zeng, and N. Fazeli. Virdo: Visio-tactile implicit representations of deformable objects. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 3583–3590. IEEE, 2022.

[127] 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.

[128] Philipp Wu, Yide Shentu, Zhongke Yi, Xingyu Lin, and Pieter Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators. *arXiv preprint arXiv:2309.13037*, 2023.

[129] Philipp Wu, Yide Shentu, Zhongke Yi, Xingyu Lin, and Pieter Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12156–12163. IEEE, 2024.

[130] Yansong Wu, Zongxie Chen, Fan Wu, Lingyun Chen, Liding Zhang, Zhenshan Bing, Abdalla Swikir, Alois Knoll, and Sami Haddadin. Tacdiffusion: Force-domain diffusion policy for precise tactile manipulation. *arXiv preprint arXiv:2409.11047*, 2024.

[131] William Xie, Stefan Caldararu, and Nikolaus Correll. Just add force for delicate robot policies. *CoRL 2024 Workshop on Mastering Robot Manipulation in a World of Abundant Data*, 2024.

[132] Haoyu Xiong, Russell Mendonca, Kenneth Shaw, and Deepak Pathak. Adaptive mobile manipulation for articulated objects in the open world. *arXiv preprint arXiv:2401.14403*, 2024.

[133] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. *arXiv preprint arXiv:2401.10891*, 2024.

[134] Sizhe Yang, Yanjie Ze, and Huazhe Xu. Movie: Visual model-based policy adaptation for view generalization. *Advances in Neural Information Processing Systems*, 36, 2024.

[135] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. *arXiv preprint arXiv:2309.13101*, 2023.

[136] Y. Ze, G. Yan, Y.-H. Wu, A. Macaluso, Y. Ge, J. Ye, N. Hansen, L. E. Li, and X. Wang. Multi-task real robot learning with generalizable neural feature fields. In *Proceedings of the 7th Annual Conference on Robot Learning*, pages StartPage–EndPage, Location of the Conference, 2023. Publisher of the Proceedings, if available. Optional note, such as a DOI or a URL if the paper is available online.

[137] Yanjie Ze, Nicklas Hansen, Yinbo Chen, Mohit Jain, and Xiaolong Wang. Visual reinforcement learning with self-supervised 3d representations. *IEEE Robotics and Automation Letters*, 8(5):2890–2897, 2023.

[138] Yanjie Ze, Ge Yan, Yueh-Hua Wu, Annabella Macaluso, Yuying Ge, Jianglong Ye, Nicklas Hansen, Li Erran Li, and Xiaolong Wang. Gnfactor: Multi-task real robot learning with generalizable neural feature fields. In *Conference on Robot Learning*, pages 284–301. PMLR, 2023.

[139] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy. *arXiv preprint arXiv:2403.03954*, 2024.

[140] Clark Zhang, Jinwook Huh, and Daniel D Lee. Learning implicit sampling distributions for motion planning. in 2018 ieee. In *RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3654–3661, 2018.

[141] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *Robotics: Science and Systems*, 2023.

[142] 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.

[143] L. Zhu, A. Mousavian, Y. Xiang, H. Mazhar, J. van Eenbergen, S. Debnath, and D. Fox. Rgb-d local implicit function for depth completion of transparent objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4649–4658, 2021.

[144] Qi Zhu, Jing Du, Yangming Shi, and Paul Wei. Neurobehavioral assessment of force feedback simulation in industrial robotic teleoperation. *Automation in Construction*, 126:103674, 2021.

[145] Matt Zucker, Nathan Ratliff, Anca D Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M Dellin, J Andrew Bagnell, and Siddhartha S Srinivasa. Chomp: Covariant hamiltonian optimization for motion planning. *The International journal of robotics research*, 32(9-10):1164–1193, 2013.