Towards Generalizable Robotic Policies via Data-Driven Learning at Scale

Jiahui Yang

CMU-RI-TR-25-65

August 26, 2025

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

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics.

Abstract

To enable robots to operate seamlessly in complex, real-world environments, they must master fine-grained manipulation skills and exhibit robust, adaptive behavior across diverse environments. This thesis explores a data-driven approach to learning generalizable and reactive manipulation policies by leveraging efficient data generation pipelines and expressive neural models. We first introduce BiDex, a low-cost teleoperation system for collecting high-quality demonstrations on dexterous bimanual tasks, addressing the challenge of acquiring real-world data. To overcome the limitations of scale and diversity in physical data collection, we present Neural MP, a simulation-based framework for autonomous data generation. Building on this foundation, we propose DRP, a learning paradigm that augments offline imitation learning with online fine-tuning and reactive components, enabling policies to perform reliably in dynamic and partially observable settings. Together, these contributions provide a practical recipe for developing robust robotic manipulation systems at scale.

Acknowledgments

I am deeply grateful to my advisor, Deepak, who trusted me and guided me into the world of robot learning. When I jointed the lab two years ago, I had no background, no prior experience, nothing in this field, just full of curiosity and a desire to learn. Deepak was fully aware of that, yet he still welcomed me into the group, patiently mentored along the way, and shaped me into the researcher I am today. Thank you for taking me on as your student and I look forward to our relationship moving forward!

I would also like to thank Russ, who has also been advising my neural motion planning series since the very beginning of my master's. Every conversation with him sharpened my thinking and deepened my appreciation for research. Looking ahead, we will work together to extend this pipeline with dexterous hand manipulation, and I am truly excited to see what the next year will bring!

I extend my heartfelt thanks to Kenny, my great mentor, friend, and collaborator. BBBBBig K is not only deeply knowledgeable, but also generous with his time, always ready to help others and provide unwavering support for us. Thanks to Jason, for sharing his energy and passion with me. It is the special JJ energy that makes him a unique person and turned the time we spent together into treasured and memorable moments.

To all my co-authors, labmates and friends—Murtaza, Youssef, Yulong, Mohan, Haoyu, Gayry, Andrew, Tony, Lili, Shagun, Jayesh, Sandeep, Russel, Unnat, Tal, Alex, Mihir, Ananye, Jianren, Chen, Sungjae—thank you for your support, collaboration, and friendship throughout this journey. It has been a true pleasure to work with and live alongside all of you.

Last but not least, I am deeply thankful to my parents for their unconditional love, support, and sacrifices. Though they know little about what robot learning is or what exactly I am doing, they have always said "yes" to whatever path I wanted to pursue. They have always been my greatest source of courage and confidence to aim high and work persistently toward my goals.

Funding

This work was supported in part by ONR MURI N00014-22-1-2773, ONR MURI N00014-24-1-2748, and Apple Research Award.

Contents

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

List of Figures

List of Tables

Chapter 1

Introduction

For robots to operate in our daily lives, they must learn fine-grained manipulation skills—the ability to precisely move, grasp, and interact with objects in unstructured, cluttered environments. Achieving this level of dexterity requires not only robust hardware but also visuomotor policies that can interpret raw sensory inputs, react to dynamic changes, and generalize across a wide range of manipulation settings.

In recent years, data-driven learning has transformed fields such as computer vision and natural language processing by leveraging large-scale datasets and expressive neural models. Inspired by this success, our work explores how similar paradigms can be applied to robotic manipulation, where learning from data offers a promising path toward scalable, generalizable, and reactive control policies.

Despite this promise, a key question remains: where does the data come from?

In order for robots to perform meaningful tasks in the real world, the most relevant data comes from real-world robot interactions themselves. However, collecting such data—particularly for dexterous, high-DOF systems—is nontrivial. In Chapter 2, we introduce a lightweight, low-cost teleoperation setup that enables high-quality data collection for dexterous bimanual tasks.

Yet even with proper tools in place, real-world data collection is inherently time-consuming and expensive, making it difficult to reach the scale and diversity required for general-purpose manipulation. To address this, we turn to simulation as a scalable alternative. In Chapter 3, we present a framework for large-scale, autonomous data generation in simulation, leveraging programmatic scene construction and expert planners to train neural motion policies. This approach enables broad generalization across tasks and environments, while significantly reducing the cost and effort of data collection.

With this, we alleviate the data bottleneck and enable scalable policy training. However, challenges still remain in the learning pipeline itself. Imitation learning in general suffers from well-known limitations: compounding errors during deployment and an upper bound on performance tied to the quality of the expert. In Chapter 4, we address these issues by introducing a learning framework that augments pretrained neural motion policies with online fine-tuning and reactive control components, enabling robust behavior in dynamic, partially observable environments.

Lastly, Chapter 5 concludes the thesis with a summary of findings and presents potential future directions to further extend this recipe.

For robots to operate in our daily lives, they must learn fine-grained manipulation skills—the ability to precisely move, grasp, and interact with objects in unstructured, cluttered environments. Achieving such level of dexterity requires not only robust hardware, but also visuomotor policies that can interpret raw sensory inputs, react to dynamic changes, and generalize across a wide variety of manipulation environments.

Chapter 2

BiDex: Learning from Real-World 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 [4, 42, 48, 58, 62]. 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

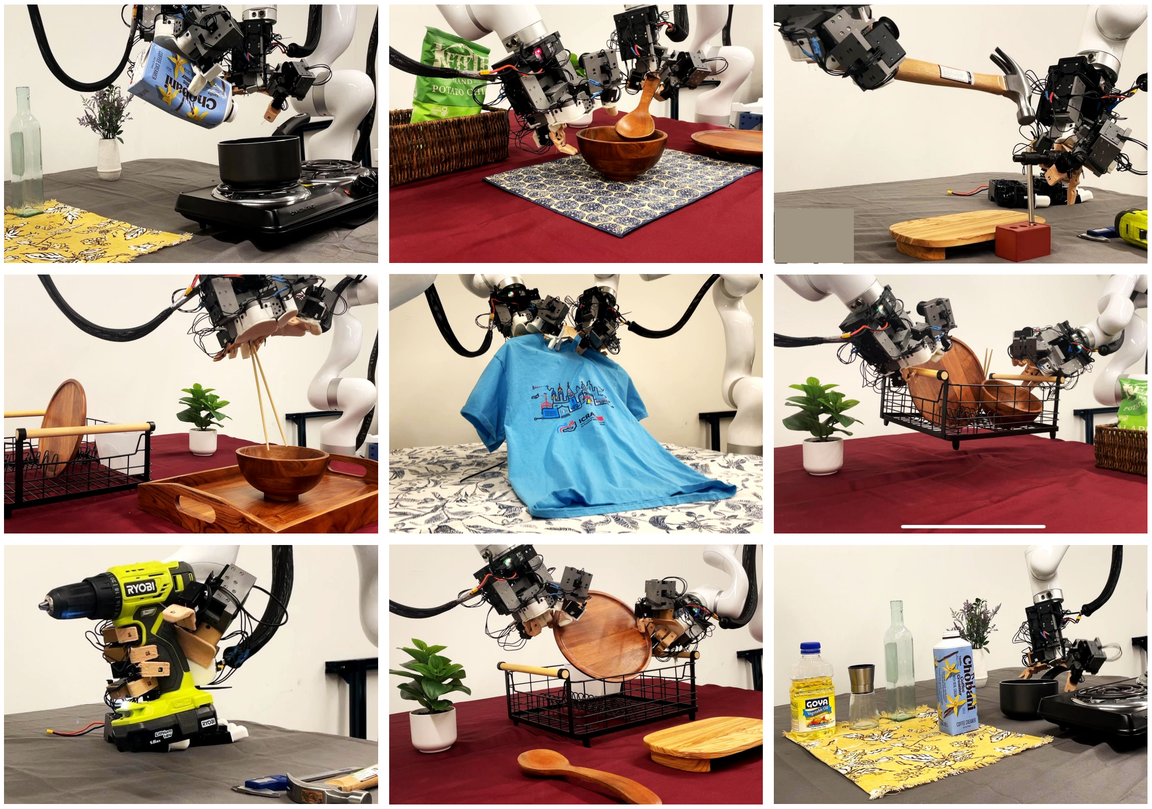
Figure 2.1: **Bimanual Dexterity**: BiDex can effortlessly teleoperate various complex tasks including pouring, scooping, hammering, chopstick picking, hanger picking, picking up basket, drilling, plate pickup and pot picking to train high-quality behavior cloning policies with over 50 degrees of freedom.

to their easy-to-use internal body tracking systems [29, 75]. However we find that wrist tracking is often jittery and the finger tracking is inaccurate. To mitigate this, SteamVR [106] 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[108] 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 [70, 98, 99, 110] for hand tracking which records human fingertip position using EMF sensors. We use the accuracte Manus Meta glove as part of our system. [71]

For arm tracking, researchers in the robotics community have been recently using joint-level teleoperation for 2-finger grippers [114, 117]. Wu et al. [114]. 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 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 [114] 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

Robot Arm Teleop Many common approaches to teleoperation include using joysticks, space mouses [67], vision-based methods, [87, 100], and VR headsets [5, 29, 75] which control arms with inverse kinematics. Joint based teleoperated control has been used in areas such as kinesthetic teaching, Brantner and Khatib [9], ABB YuMi [63] and Da Vinci Machines [34], and recently [35, 118] introduced a low-cost version of this with mirroring Trossen Robot arms. GELLO [114] 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

[37, 81, 100] 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 [96]. Recently, bimanual robot hands and tracking have become accessible for academic labs. Dexcap uses LEAP Hand [98] and tracks the human with gloves and a SLAM-based robot camera. [110] Hato uses a VR headset and controller to control two 6 DOF Psyonic Hands [66, 78]. 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 [37, 81, 100, 110].

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. [106] Vicon-based systems use reflective balls and external cameras to track. Inside-out approaches such as XSens [73] or Rokoko suit rely on IMUs on the body but these often drift over time and require recalibration [66, 89]. For hand data, many vision-based approaches such as Frankmocap [91] return MANO [90] 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 [20, 69]. Complementing these advances, significant efforts have been made to scale up robotic datasets to facilitate more capable robotic systems [25, 55, 74, 109]. 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 [21, 35, 101, 118]. 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, even those with over 20 degrees of freedom. It achieves accurate tracking of the human hand using a Manus VR glove-based system [71] and human arm tracking through a GELLO-inspired system [114]. We outline the process for sending commands and collecting data with bimanual hands in Alg.**??**. Importantly, our solution functions seamlessly in both tabletop and mobile environments, as it requires no external tracking devices and is highly portable. In Section 3.4, 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 [98] and LEAP Hand V2 [97].

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 [76, 91] or VR headsets, often face significant inaccuracies due to occlusions and varying lighting conditions, as discussed in Section 3.4. 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 without hindering movement. In BiDex, we opt for the Manus Glove [71], which has demonstrated reliable tracking performance without overheating or suffering from calibration problems as seen with alternatives like the Rokoko gloves [66, 89]. 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

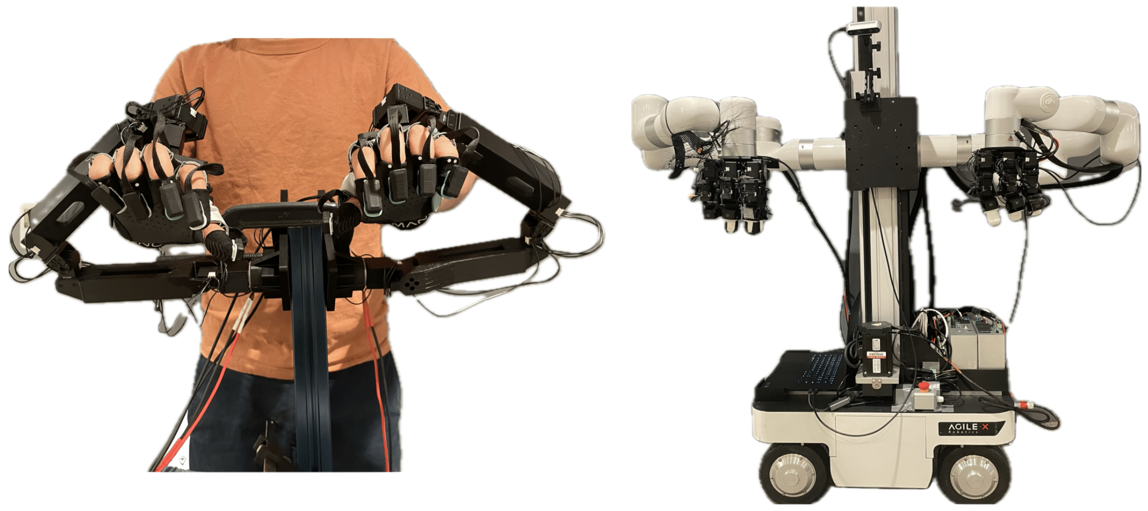

Figure 2.2: **Mobile bimanual teleoperation system** Left: An operator strapped into BiDex. Right: Our bimanual robot setup including two xArm robot arms, two LEAP Hands [98] and three cameras on an AgileX base.

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 [37, 100] has addressed this challenge by ensuring that pinch grasps are consistent between human and robot hands. Wang et al. [110] 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 [12]. We employ the Manus gloves [71] 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 Wu et al. [114], Zhao et al. [117] 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 from Wu et al. [114] 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

Tabletop Manipulation For our tabletop setup, the robotic arms are positioned to face each other similar to Zhao et al. [117] while the GELLO teaching arms mirror this configuration. Compared to a side-by-side configuration like in Wang et al. [110], 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

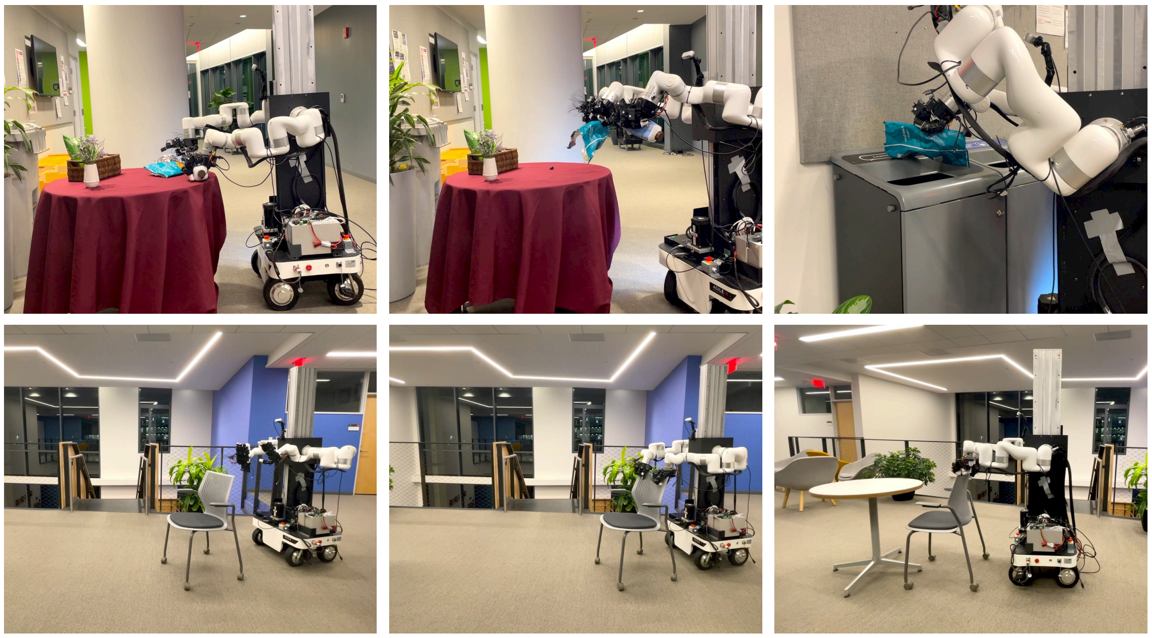
Figure 2.3: **All Tasks**: Teleoperation of the mobile robot systems with BiDex. **Top**: Picking up trash from a table and discarding it into a bin. **Bottom**: Grasping a chair and moving it to align with a table.

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 [3, 115]. A secondary operator uses a joystick to control the mobile base and manage task resets.

2.3 Experiment Setup

2.3.1 Baseline Teleoperation Approaches

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 [29, 75] As a baseline we use the Apple Vision Pro which returns both finger data similar to MANO parameters [76] 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 [96] and academia alike. [2, 70] 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

Leap Hand LEAP Hand, introduced by Shaw et al. [98] 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 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

Tabletop 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.

Mobile 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.

2.4 Results

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 by Shaw et al. [98] 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.

	Completion Rate			Time Taken		
	Handover	Cup Stacking	Bottle Pouring	Handover	Cup Stacking	Bottle Pouring
`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 one baseline method, SteamVR. BiDex enables more reliable and faster data collection, especially for harder tasks like bottle pouring.

2.4.1 Bimanual Dexterous Teleoperation Results

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

	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 [98], for different tasks. BiDex is versatile and compact enough to be adopted to successfully collect data for mobile tasks.

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.

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.

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

Table 2.3: **Imitation learning**: We train ACT from [117] 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.

2.4.2 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 [117] with a horizon length of 16 at 30hz using pretrained weights from [26] 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 [13] handover, we even see good generalization of the policy to different initial locations of the can.

2.4.3 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.

	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.

To do this, we show a variety of very challenging tasks as in Figure 4.1 and Figure 2.4. These tasks include pouring, scooping, hammering, chopstick picking, hanger picking, 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`

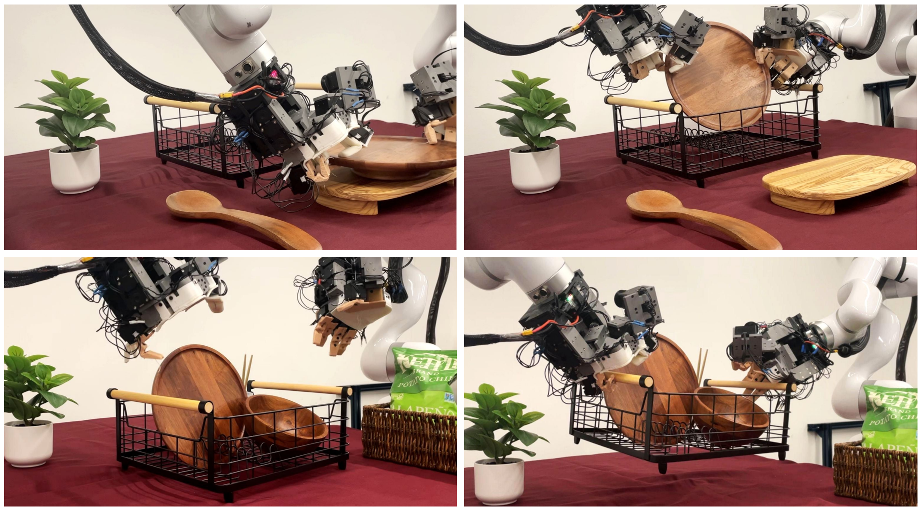

Figure 2.4: **Clearing the Dishes**: In this task, we use BiDex to perform a long horizon task to place bowls and spoons into a drying rack and lift the drying rack away from the table.

2.5 Discussion and Limitations

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

Neural MP: Learning from Large-Scale Synthetic Dataset

Motion planning is a longstanding problem of interest in robotics, with previous approaches ranging from potential fields [54, 113], sampling (RRTs and Roadmaps) [49, 51, 57, 61, 103], search (A*) [39, 56, 64] and trajectory optimization [30, 95, 105].

Despite being ubiquitous, these methods are often slow at producing solutions since they largely plan from scratch at test time, re-using little to no information outside of the current problem and what is engineered by a human designer. Since motion-planning is a core component of the robotics stack for manipulation, its speed, capability and ease of use form a core bottleneck to developing efficient and reliable manipulation systems.

On the other hand, humans can generate motions in a closed loop manner, move quickly, react to various dynamic obstacles, and generalize across a wide distribution of problem instances.

Rather than planning open loop from scratch, people draw on their vast amounts of experience moving and interacting with their environment while reactively adjusting their movements in order to quickly and efficiently move about the world. How can we create motion planners with similar properties?

In this work, we argue that *distillation* at scale is the answer: we can *distill* the planning process into a reactive neural policy.

The primary challenge in training data-driven motion planning is the data collec-

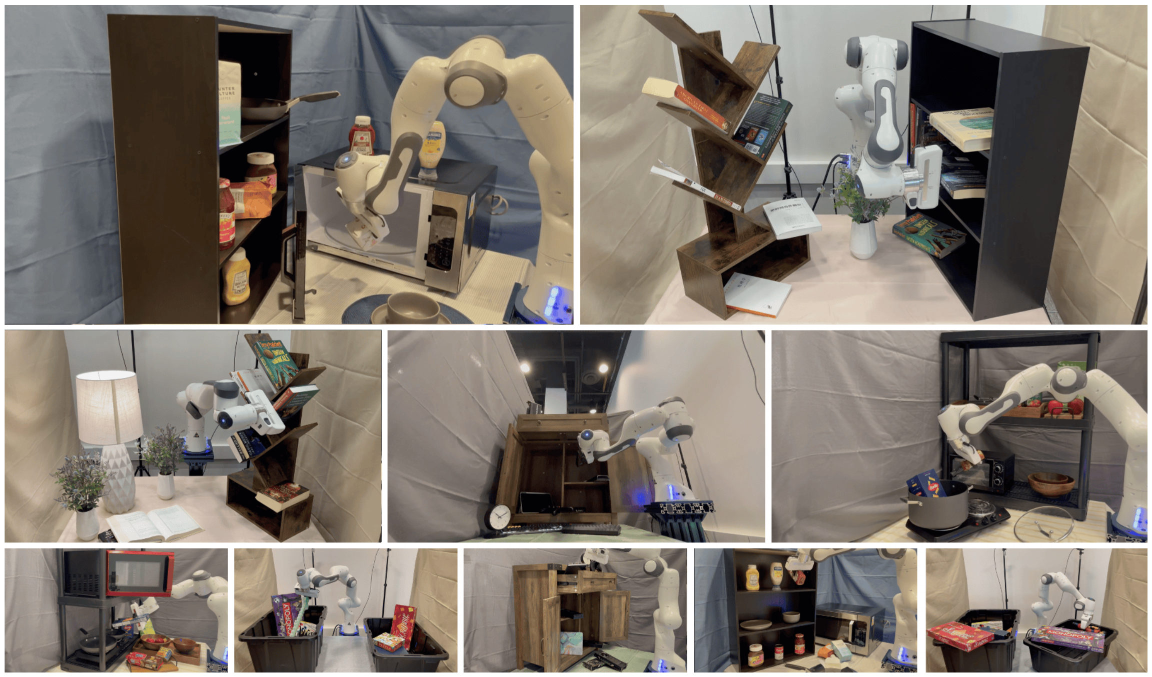

Figure 3.1: **Neural Motion Planning at Scale in the Real World** Our approach enables a *single* neural network policy to solve motion planning problems across diverse setups; Neural MP can generate collision free motions for a wide array of unseen tasks significantly faster and with higher success rates than traditional as well as learning-based motion planning approaches.

tion itself, as scaling robotic data collection in real-world requires significant human time and effort. Recently, there has been a concerted effort to scale up data collection for robot tasks [55, 74]. However, the level of diversity of scenes and arrangement of objects is still limited, especially for learning obstacle avoidance behavior that scales to the real world. Constructing such setups with diverse obstacle arrangements with numerous objects is prohibitively expensive in terms of cost and labor.

Instead, we leverage simulation, which makes it cheap and easy to obtain diverse data, is highly scalable via parallelization, and runs significantly faster than real world. Recent approaches have shown great promise in enabling policy learning for high-dof robots [1, 19]. We build a large number of complex environments by combining procedural, programmatic assets with models of everyday objects sampled from large 3D datasets. These are used to collect expert data from state-of-the-art (SOTA) motion planners [103], which we then *distill* into a reactive neural policy.

Since this policy has seen data from **1 million** scenes, it is capable of generalizing to novel obstacles and scene configurations that it has never seen before. However, deploying neural policies in the real world might be unsafe for the system due to the potential of collisions. We mitigate this by using a linear model to predict future states the robot will end up in and run optimization to ensure a safe path.

Our core contribution is a SOTA motion planner that runs zero-shot on any environment, with more accuracy and in orders of magnitude less execution time.

We demonstrate that large scale data generation in simulation can enable training generalizable policies that can be successfully deployed for real-world motion planning tasks. To our knowledge, Neural MP is the first work to demonstrate that such a neural policy can generalize to a broad set of out-of-distribution of real-world environments, generalizing across tasks with significant variation across poses, objects, obstacles, backgrounds, scene arrangements, in-hand objects, and start/goal pairs. Specifically, we propose a simple, scalable approach for training and deploying fast, generalizable neural motion planners: 1) **large-scale procedural scene generation** with diverse environments in realistic configurations, 2) **multi-modal sequence modeling** for fitting to sampling-based motion planning data and 3) **lightweight test-time optimization** to ensure fast, safe, and reliable deployment in the real world.

We execute a thorough real-world empirical study of motion-planning methods, evaluating our approach on **64** real world motion planning tasks across **four** diverse environments, demonstrating a motion planning success rate improvements of **23%** over sampling-based, **17%** over optimization-based and **79%** over neural motion planning methods. Video results available at mihdalal.github.io/neuralmotionplanner.

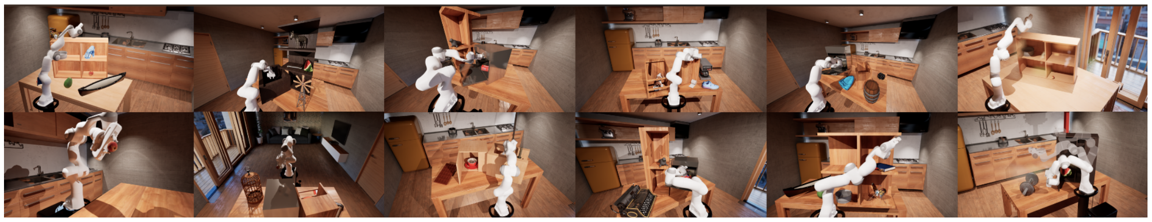

Figure 3.2: **Visualization of Diverse Simulation Training Environments**: We train Neural MP on a wide array of motion planning problems generated in simulation, with significant pose, procedural asset, and mesh configuration randomization to enable generalization.

3.1 Related Work

Approaches for Training General-Purpose Robot Policies Prior work on large scale imitation learning using expert demonstrations [10, 11, 23, 55, 74] has shown that large models trained on large datasets can demonstrate strong performance on challenging tasks and some varying levels of generalization. On the other hand, sim2real transfer of RL policies trained with procedural scene generation has demonstrated strong capabilities for producing generalist robot policies in the locomotion regime [1, 19].

In this work, we combine the strengths of these two approaches to produce powerful neural motion planning policies. We propose a method for procedural scene generation in simulation and combine it with large scale imitation learning to produce strong priors which we transfer directly to over 64 motion planning problems in the real world.

Procedural Scene Generation for robotics Automatic scene generation and synthesis has been explored in vision and graphics [88, 111] while more recent work has focused on embodied AI and robotics settings [23, 27, 50, 112]. In particular, methods such as Robogen [112] and Gen2sim [50] use LLMs to propose tasks and build scenes using existing 3D model datasets [28] or text-to-3D [77] and then decompose the tasks into components for RL, motion-planning and trajectory optimization to solve in simulation. Our method is instead rule-based rather than LLM-based, and is designed specifically for generating data to train neural motion planners (see Sec. 3.2.1), and demonstrates that policies trained on its data can indeed be transferred to the real world. MotionBenchmaker [17], on the other hand, is similar to our data generation method in that it autonomously generates scenes using programmatic assets. However, the datasets generated by MotionBenchmaker are not realistic: floating robots, a single major obstacle per scene and primitive objects that are spaced far apart. By comparison, the scenes and data generated by our work (Fig. 3.2) are considerably more diverse, containing additional programmatic assets that incorporate articulations (microwave, dishwasher), multiple large obstacles per scene (up to 5), complex meshes sampled from Objaverse [28], and tightly packed obstacles.

Neural Motion Planning Finally, there is a large body of recent work [14, 31, 41, 44, 83, 85, 93] focused on imitating motion planners in order to accelerate planning.

MPNet [47, 83, 85] trains a network to imitate motion planners, then integrates this prior into a search procedure at test time. Our method leverages large scale scene generation and sequence modeling, enabling it to use a faster optimization process at test time while obtaining strong results across a diverse set of tasks.

MπNets [31] trains the SOTA neural motion planning policy using procedural scene generation and demonstrates transfer to the real world. Our approach is similar, albeit with 1) much more diverse scenes via programmatic asset generation and complex real-world meshes, 2) a more powerful learning architecture and multi-modal output distributions and 3) test-time optimization to improve performance at deployment, enabling significantly improved performance over MπNets.

3.2 Neural Motion Planning

Our approach enables generalizable neural motion planners, by leveraging large amounts of training data generated in simulation via expert planners. The policies can generalize to out-of-distribution settings by using powerful deep learning architectures along with diverse, large-scale training data. To further improve the performance of these policies at deployment, we leverage test time optimization to select the best path out of a number of options. We now describe each of these pieces in more detail.

3.2.1 Large-scale Data Generation

One of the core lessons of the deep learning era is that the quality and quantity of data is crucial to train broadly capable models. We leverage simulation to generate vast datasets for training robot policies. Our approach generates assets using programmatic generation of primitives and by sampling from diverse meshes of common objects. These assets are combined to create complex scenes resembling real world scenarios (Fig. 3.2), as described in Alg. 1.

Procedural Generation From Primitives How do we generate a large enough number of diverse environments to train a generalizable policy? Hand designing each environment is tedious, requiring significant human effort per scene, which doesn't scale well. Instead, we take the approach of *procedural* scene generation, using a set of six *parametrically variable* categories - shelves, cubbies, microwaves,

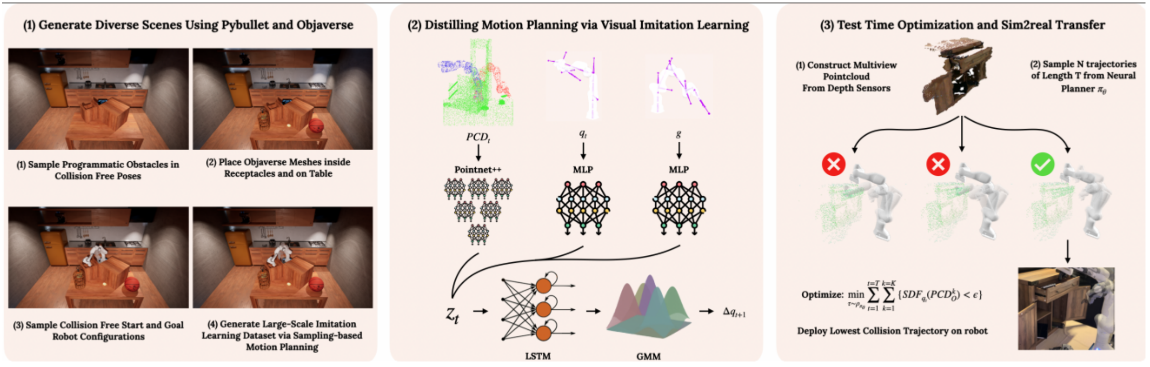
(1) Generate Diverse Scenes Using Pybullet and Objaverse
(1) Sample Programmatic Obstacles in Collision Free Poses
(2) Place Objaverse Meshes inside Receptacles and on Table
(3) Sample Collision Free Start and Goal Robot Configurations
(4) Generate Large-Scale Imitation Learning Dataset via Sampling-based Motion Planning
(2) Distilling Motion Planning via Visual Imitation Learning
PCD_t
q_t
g
Pointnet++
MLP
MLP
z_t
LSTM
GMM
Δq_{t+1}
(3) Test Time Optimization and Sim2real Transfer
(1) Construct Multiview Pointcloud From Depth Sensors
(2) Sample N trajectories of Length T from Neural Planner π_θ
Optimize: $\min_{\tau \sim \rho_{\pi_\theta}} \sum_{t=1}^{t=T} \sum_{k=1}^{k=K} \{SDF_{q_t}(PCD_O^k) < \epsilon\}$
Deploy Lowest Collision Trajectory on robot

Figure 3.3: **Method Overview**: We present Neural Motion Planners, which consists of 3 main components. **Left**: Large Scale data generation in simulation using expert planners **Middle**: Training deep network models to perform fast reactive motion planning **Right**: Test-time optimization at inference time to improve performance.

dishwashers, open boxes, and cabinets. These categories are representative of a large set of objects in everyday scenarios that robots encounter and have to avoid colliding with. Each category instance is constructed using a combination of primitive cuboid objects and is parameterized by category specific parameters which define the asset. Specifically a category instance g is comprised of N cuboids $g = \{x_0..x_i...x_N\}$, which satisfy the category level constraint given by $C(g)$. For controlled variation within each category, we make use of *parametric* category specific generation functions $X(\mathbf{p}) = \{x_0..x_i.x_N\}$, s.t. $C(X(\mathbf{p}))$, where p specifies the size and scale of each of the cuboids, their relative positions, and specific axes of articulation. The constraint C(.) relates to the relative positions, scales and orientations of the different cuboids, *e.g* for the microwave category the constraint ensures each of the walls are of the same height, and that the microwave has a hinge door.

Objaverse Assets For Everyday Objects While programmatic generation can create a large number of scenes using the defined categories, there are a large number of everyday objects the robot might encounter that lie outside this distribution. For example, a robot will need to avoid collisions with potted plants, bowls and utensils while moving between locations, as shown in Fig 4.1.

To better handle these settings, we augment our dataset with objects sampled from the recently proposed large-scale 3D object dataset, Objaverse [28]. This dataset contains a wide variety of objects that the neural planner is likely to observe during deployment, such as comic books, jars, record players, caps, etc. We sample these

Algorithm 1 Procedural Scene Generation

```
Require: Asset category generators $\{X_i(\mathbf{p})\}_{0,1..G}$
Require: Number of scenes $N$
Require: Max objects per scene $K$
Require: Collision checker $Q$
 1: for scene 1: N do
 2:     Initialize scene $S = \{\}$
 3:     Sample number of assets $k \sim [1, ...K]$
 4:     for asset 1:k do
 5:         Sample asset category $g \sim [0, ..N]$
 6:         Sample asset parameter $p$
 7:         Sample asset $x \sim X_g(p)$
 8:         while $Q(S, x)$ do
 9:             for each asset $s_i$ in $S$ do
10:                 $n_i$ = collision normal b/n $x$ and $s_i$
11:             end for
12:             Effective collision normal $n = \sum n_i$
13:             Update $p$ so $X_g(p)$ center is shifted along $n$
14:         end while
15:         Add asset $x$ to scene $S$
16:     end for
17:     yield scene S
18: end for
```

Objaverse assets in the task-relevant sampling location of the programmatic asset(s) in the scene, such as between shelf rungs, inside cubbies or within cabinets.

Complex Scene Generation The scenes we use comprise combinations of the procedurally generated assets built from primitives, and the Objaverse assets arranged on a flat tabletop surface. A naive approach to constructing realistic scenes is to use rejection-sampling based on collision. This involves iteratively sampling assets on a surface, and re-sampling those that collide with the current environment. However, as the number, size and type of objects increases, so does the probability of sampling assets that are in collision, making such a process prohibitively expensive to produce a valid configuration. In addition, this is biased towards simple scenes with few assets that are less likely to collide, which is not ideal for training generalizable policies. Instead, we propose an approach that iteratively adds assets to a scene by adjusting their position using the effective collision normal vector, computed from the existing assets in the scene. Please see Alg. 1 for additional details.

Motion Planning Experts:

To collect expert data in the diverse generated scenes, we leverage SOTA sampling-based motion planners due to their (relative) speed as well as ease of application to a wide array of tasks. Specifically, we use Adaptively Informed Trees [103] (AIT*), an almost-surely asymptotically optimal sampling-based planner to produce high-quality plans using privileged information, namely access to a perfect collision checker in simulation.

How do we ensure that the planner is evaluated between points in the scene that require it to maneuver around obstacles?

We generate tight-space configurations by sampling end-effector poses from specific locations (*e.g.*, inside a cubby or microwave) and by using inverse kinematics (IK) to derive the joint pose. Tight-space configurations are sampled 50% of the time, to ensure that we collect trajectories where the robot moves around obstacles, as opposed to taking straight line paths between nearby free space points.

Additionally, we spawn objects grasped in the end-effectors, with significant randomization including boxes, cylinders, spheres or even Objaverse meshes. Importantly, we found that naively imitating the output of the planner performs poorly in practice as the planner output is not well suited for learning. Specifically, plans produced by AIT* often result in way-points that are far apart, creating large action jumps and sparse data coverage, making it difficult for networks to fit the data. To address this issue, we perform smoothing using cubic spline interpolation while enforcing velocity and acceleration limits. We found that smoothing is crucial for learning performance as it ensures action size limits for each time-step transition.

3.2.2 Generalizable Neural Policies

We would like to obtain agents that can use diverse sets of experiences to plan efficiently in new settings. In order to build generalizable neural motion planning policies, we need an observation space amenable to sim2real transfer, and utilize an architecture capable of absorbing vast amounts of data.

Observations: We begin by addressing the sim2real transfer problem, which requires considering the observation and action spaces of the trained policy. With regards to observation, point-clouds are a natural representation of the scene for transfer [18, 22, 31, 45], as they are 3D points grounded in the base frame of the robot and therefore view agnostic, and largely consistent between sim and real. We include proprioceptive and goal information in the observations, consisting of the current joint angles q_t, the target joint angles g, in addition to the point-cloud PCD.

Network Architecture: We require an architecture capable of scaling with data while performing well on multi-modal sequential control problems, *e.g.* motion planning. To that end, we design our policy π (visualized in Fig. 4.2) to be a sequence model to imitate the expert using a notion of history which is useful for fitting privileged experts using partially observed data [23]. In principle, any sequence modeling architecture could be used, but in this work, we opt for LSTMs for their fast inference time and comparable performance to Transformers on our datasets (see Appendix). We operate the LSTM policy over joint embeddings of PCD_t, q_t, and g with a history length of 2.

We encode point-clouds using PointNet++ [80], while we use MLPs to encode q_t and g_t. We follow the design decisions from MπNets regarding point-cloud observations: we segment the robot point-cloud, obstacle point-cloud and the target robot point-cloud before passing it to PointNet++. For each time-step, we concatenate the embeddings of each of the observations together into one vector and then pass them into the LSTM for action prediction.

For the output of the model, note that sampling-based motion planners such as AIT* are heavily multi-modal: for the same scene they may give entirely different plans for different runs. As a result, we require an expressive, multi-modal distribution to effectively capture such data, for which we use a Gaussian Mixture Model (GMM). Specifically, Neural MP predicts a GMM distribution over delta joint angles (Δq_{t+1}), which are used to compute the next target joint way-point during deployment: $q_{t+1} = q_t + \Delta q_{t+1}$.

As we show in our experiments, for fitting to sampling-based motion planning, minimizing the negative log-likelihood of the GMM outperforms the PointMatch loss from MπNets, Diffusion [93] and Action-chunking [118] (Sec. 3.4).

3.2.3 Deploying Neural Motion Planners

Test time Optimization While our base neural policy is capable of solving a wide array of challenging motion planning problems, we would still like to ensure that these motions are safe to be deployed in real environments. We enable this property by combining our learned policy with a simple light-weight optimization procedure at inference time.

This relies on a simple model that assumes the obstacles do not move and the controller can accurately reach the target way-points. Given world state $s = [q, e]$ (e is the environment state), the predicted world state is $s' = [q + \hat{a},$ e] where $\hat{a}$ is the policy prediction. With this forward model, we can sample N trajectories from the policy using the initial scene point-cloud to provide the obstacle representation and estimate the number of scene points that intersect the robot using the linear forward model. We then optimize for the path with the least robot-scene intersection in the environment, using the robot Signed Distance Function (SDF). Specifically, we optimize the following objective at test time:

$$\min_{\tau \sim \rho_{\pi_\theta}} \sum_{t=1}^{t=T} \sum_{k=1}^{k=K} 1\{SDF_{q_t}(PCD_O^k) < \epsilon\} \tag{3.1}$$

in which ρ_{π_θ} is the distribution of trajectories under policy π_θ with a linear model as described above, PCD_O^k is the kth point of the obstacle point-cloud (with max

$K = 4096$ points) and SDF_{q_t} is the SDF of the robot at the current joint angles. In practice, we optimize this objective with finite samples in a single step, computing the with minimal objective value by selecting the path with minimal objective value across 100 trajectories.

Sim2real and Deployment For executing our method on a real robot, we predict delta joint way-points which we then linearly interpolate and execute using a joint space controller. Our setup includes four extrinsically calibrated Intel RealSense cameras (two 435 and two 435i) positioned at the table's corners.

To produce the segmented point cloud for input to the robot, we compute a point-cloud of the scene using the 4 cameras, segment out the partial robot cloud using a mesh-based representation of the robot to exclude points.

We then generate the current robot and target robot point clouds using forward kinematics on the mesh-based representation of the robot and place them into the scene. For real-world vision-based collision checking, we calculate the SDF between the point cloud and the spherical representation of the robot, enabling fast SDF calculation (0.01-0.02s per query), though this method can lack precision for tight spaces.

3.3 Experimental Setup

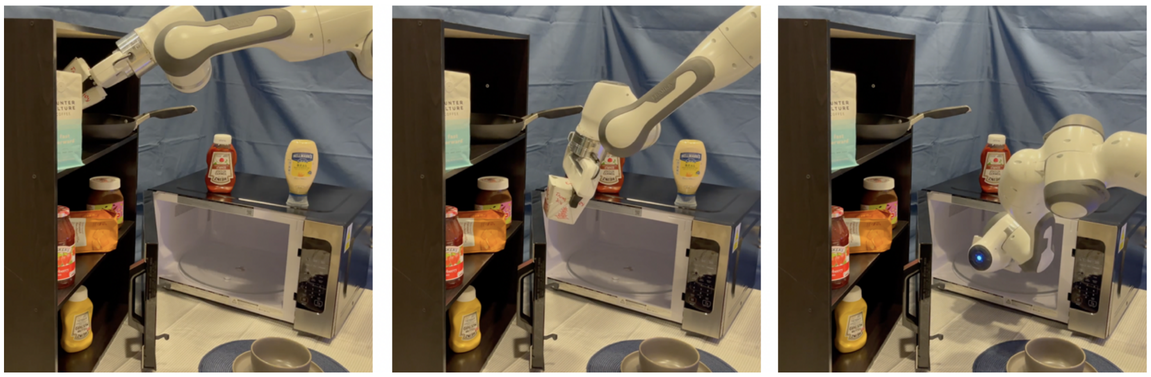

Figure 3.4: Sampling-based planners struggle with tight spaces, a regime in which Neural MP performs well.

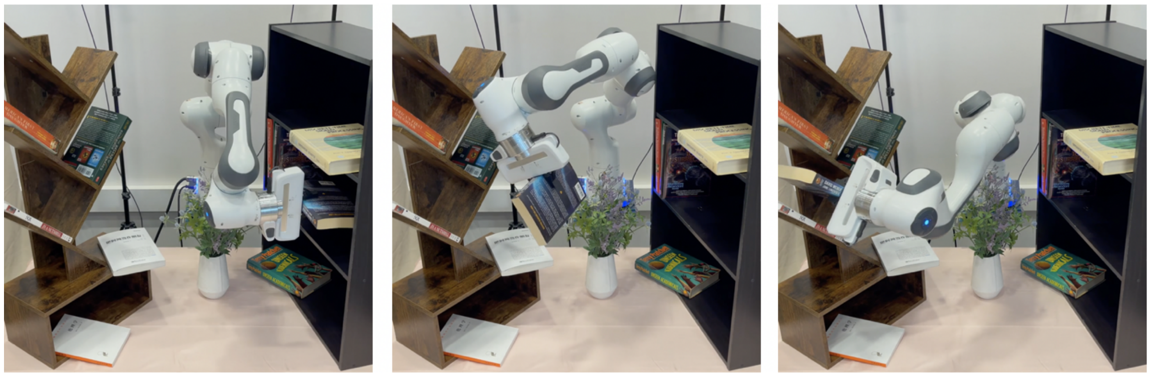
Figure 3.5: Our policy has not been trained on this bookcase, yet it is able to insert the book into the correct location.

In our experiments, we consider motion planning in four different real world environments containing obstacles (check our project website). Importantly, these are not included as part of the training set, and thus the policy needs to generalize to perform well on these settings. We begin by describing our environment design, then our evaluation protocol and comparisons.

Environment Design

We evaluate our motion planner on tabletop motion planning tasks which we subdivide into *environments*, *scenes*, and *configurations*. We evaluate on four different environments, with each environment containing 1-2 large receptacles that function as the primary obstacles. For each environment, we have four different scenes which involve significant pose variation (over the entire tabletop) of the primary obstacles, table height randomization, as well as randomized selection, pose and orientation of objects contained within the receptacles. For each environment, we have two scenes with obstacles and two without obstacles. For each scene, we evaluate on four different types of start (q_0) and goal (g) angle pairs: 1) free space to free space, 2) free space to tight space 3) tight space to free space 4) tight space to tight space.

Free space configurations do not have an obstacle in the vicinity of the end-effector, while tight space configurations generally have obstacles on most sides of the end-effector.

Our four environments are 1) **Bins**: moving in-between, around and inside two different industrial bins 2) **Shelf**: moving in-between and around the rungs of a black shelf 3) **Articulated**: moving inside and within cubbies, drawers and doors 4) **in-hand**: moving between rungs of a shelf while holding different objects.

Evaluation Protocol We evaluate all methods on open loop planning performance for fairness, though our method, just like MπNets, is capable of executing trajectories in a closed loop manner. For neural planners such as our method and MπNets, this involves generating an open loop path by passing the agent's predictions back into itself using a linear model for the next state, as described in Sec. 3.2.3. We then execute the plans on the robot, recording the *success rate* (SR) of collision free goal reaching and the robot's *collision rate* (CR). We follow MπNets' definition of success rate: reaching within 1cm and 15 degrees of the goal end-effector pose of the target goal configuration without any collision. In practice, our policy achieves orientation errors significantly below this threshold, 2 degrees or less.

Comparisons We propose three baselines for real-world comparisons to evaluate different aspects of our method's capabilities. We compare against sampling-based motion planning, which is expensive to run but has strong guarantees on performance. The first baseline is the expert we use to train our model, AIT* with 80 seconds of planning time. We run this planner with the same vision-based collision checker used by our method in the real world. AIT*-80s is impractical to deploy in most settings due to its significant planning time. Thus, we compare to a faster variant of AIT* with 10 seconds of planning time, which uses comparable time to our method (Note: Our method takes 3s to plan, but AIT*-3s is unable to find a plan for any real world task).

Next, we compare against Curobo [105], a SOTA motion-generation method which performs GPU-parallelized optimization and is orders of magnitude faster than AIT*. We run this baseline with a voxel-based collision checker and optimize its voxel resolution per task due to its sensitivity to that parameter. Finally, we compare against the SOTA neural motion planning approach, MπNets, both with and without *test-time optimization* (TTO).

3.4 Experimental Results

To guide our evaluation, we pose a set of experimental questions. 1) Can a single policy trained in simulation learn to solve complex motion planning tasks in the real world? 2) How does Neural MP compare to SOTA neural planning, sampling-based and trajectory optimization planning approaches? 3) How well does Neural MP

	Bins	Shelf	Articulated	Avg. SR	Avg. CR
Sampling-based Planning:					
`AIT*-80s [103]`	93.75	75.0	50.0	72.92	**0**
`AIT*-10s (fast) [103]`	75.0	37.5	25.0	45.83	2.08
Optimization-based Planning:					
`Curobo [105]`	93.75	81.25	62.5	79.17	2.08
Neural:					
`M`π`Nets [31]`	18.75	25.0	6.25	16.67	18.75
`M`π`Nets (with TTO) [31]`	18.75	25.0	6.25	16.67	14.58
`Ours-Base Policy`	81.25	75.0	43.75	66.67	33.33
`Ours`	**100**	**100**	**87.5**	**95.83**	4.17

Table 3.1: Neural MP performs best across each scene free-hand motion planning task, demonstrating greater improvement as the task complexity grows.

extend to motion planning tasks with objects in-hand? 4) Can Neural MP perform dynamic obstacle avoidance? 5) What are the impacts key ingredients of Neural MP have on its performance?

Free Hand Motion Planning In this set of experiments, we evaluate motion planning while the robot's hand is empty (Table 3.1). We find that our base policy on its own performs comparably to AIT*-80s (66.67% vs. 72.92%) while only using 1s of planning time. When we include test-time optimization (3s of planning), we find that across all three tasks, Neural MP achieves the best performance with a 95.83% success rate. In general, we find that Bins is the easiest task, with the sampling/optimization-based methods performing well, Shelf is a bit more difficult as it requires simultaneous vertical and horizontal collision avoidance, while Articulated is the most challenging task as it contains a diverse set of obstacles and tight spaces. Neural MP performs well across each task as it is trained with a diverse set of objects that cover the types of real-world obstacles we encounter and it also incorporates complex meshes which cover the irregular geometries of the additional objects we include.

In our experiments, MπNets performs poorly across the board. We attribute this finding to 1) MπNets is only trained on data in which the expert goes from tight spaces to tight spaces, which means it fails to generalize well to start/goal configurations in free space and 2) the end-effector point matching loss in MπNets fails to distinguish between 0 and 180 degree rotations of the end-effector, so the network has not learned how to match ambiguous target end-effector poses. Note, even if we change the success rate metric for MπNets to count 180 degree flipped end-effector poses as successes as well, the average success rate of MπNets only improves from 16.67% to 29.17% - it is still far below the other methods. 3) MπNets is a deterministic policy, so test-time optimization yields limited improvement. Meanwhile, failure cases for AIT* and Curobo are tight spaces for which vision-based collision checking is inaccurate and the probability of sampling/optimizing for a valid path is low. In contrast, our method performs well on each task, generalizing to 48 different unseen environment, scene, obstacle and joint configuration combinations.

In-Hand Motion Planning In this experiment, we extend our evaluation to motion planning with objects in-hand, a crucial capability for manipulation. We evaluate Neural MP against running the neural policy without test time optimization and without including any Objaverse data, achieving 81% performance vs. 31% and 44%. We visualize an example trajectory in Fig. 3.4, 3.5. Our method performs well on in-distribution objects such as the book and board game, but struggles on out of distribution objects such as the toy sword, which is double the size of objects at training time.

We additionally deploy our method on significantly out of distribution objects such as the bookcase (Fig. 3.5) and find that Neural MP generalizes well to in-hand motion planning tasks such as inserting the book in the right rung.

This experiment also serves as an ablation of our method, demonstrating the importance of test time optimization on out of distribution scenarios. For these tasks, the base policy performance results in a large number of collisions as two of the in-hand objects are out of distribution (sword and board game), but the optimization step is able to largely remove them and produce clean behavior that reaches the target without colliding. Additionally, this experiment demonstrates that the Objaverse data is crucial for the success of our method in the real world. Models trained only on cuboid-based parametric assets fail to generalize to the complexity of the real world (43.75%) while those trained on Objaverse perform well (81.25%), highlighting the importance of incorporating Objaverse meshes into scene generation.

Dynamic Motion Planning In many real-world scenarios, the environment may

	Global	Hybrid	Both	Average
`MPNet [83]`				
Hybrid Expert	41.33	65.28	67.67	58.09
`M`π`Nets [31]`				
Global Expert	75.06	80.39	82.78	79.41
Hybrid Expert	75.78	95.33	95.06	88.72
`EDMP [93]`				
Global Expert	71.67	82.84	82.79	79.10
Hybrid Expert	75.93	86.13	85.06	82.37
`Ours`				
Global Expert	**77.93**	85.50	87.67	83.70
Hybrid Expert	76.33	**97.28**	**96.78**	**90.13**

Table 3.2: Comparison of neural motion planning methods across 5400 test problems in the MπNets dataset in simulation. Neural MP achieves the highest success rates on these tasks.

be changing as the motion planner is acting. We test how well Neural MP can motion plan in such settings by introducing obstacles into the environment while the motion planner is moving to a goal. We evaluate the motion planner on four different goals with three different added obstacles (drawer, monitor and pot). To handle dynamic obstacles, we run the neural motion planner closed loop and perform single-step test-time optimization. We compare against MπNets and find that Neural MP performs 53% better (63.33% vs. 10%), performing particularly well on the drawer and pot object while struggling on the monitor object which is significantly taller. We also qualitatively evaluate Neural MP on two significantly more challenging motion planning tasks in which we continuously move the obstacle into the robot's path and demonstrate that it can adjust its behavior to avoid collisions while reaching the goal.

Comparisons to Learning-based Motion Planners We next evaluate how Neural MP compares to two additional learning-based methods, MPNets [83] and EDMP [93] (a Diffusion-based neural motion planner) as well as MπNets [31] in simulation. We compare these three neural motion planning methods in simulation trained on the same dataset (from MπNets) of 3.27 million trajectories. We train policies on the Global expert data and the Hybrid datasets and then evaluate on 5400 test problems across the Global, Hybrid and Both solvable subsets. We include numerical results Tab. 3.2, with numbers for the baselines taken from the EDMP and MπNets papers. We find that across the board, Neural MP is the best learning-based motion planning method, outperforming both EDMP and MπNets. We attribute this to the use of sequence modelling, the ability of the GMM to fit multimodal data and test-time optimization to prune out any collisions.

Data Scaling In order to understand the scaling of our method with data we evaluate how performance changes with dataset size. To do so, we train models with 1K trajectories, 10K trajectories and 100K trajectories. In these experiments, we train with subsets of our overall dataset and evaluate on held out simulation environments which are not sampled from the training distribution.

While performance with a thousand trajectories is weak (15%), we find rapid improvement as we increase the orders of magnitude of data (10K - 50%, 100K - 65%), with the model trained on 1M trajectories achieving 80% success rate on entirely held out tight-space shelf and bin configurations, demonstrating that our method scales and improves with data.

Ablations We run ablations of components of our method (training objective, model architecture) in simulation to evaluate which have the most impact. For each ablation we evaluate performance on held out scenes. For training objective, we find that GMM (ours) outperforms L2 loss, L1 loss, and PointMatch Loss (MπNets) by (7%, 12%, and 24%) respectively. On the model architecture side, LSTM policy (ours) achieved comparable performance to the Transformer variant (Table 3.3), while with significantly faster inference time (2x). Hence we opt to use LSTM policies for our experimental evaluation.

Neural MP-MLP	**Neural MP-LSTM**	**Neural MP-Transformer**	**Neural MP-ACT**
65.0	82.5	**85.0**	47.5

Table 3.3: Success rates of different architecture choices.

3.5 Discussion and Limitations

In this work, we present Neural MP, a method that builds a data-driven policy for motion planning by scaling procedural scene generation, distilling sampling-based motion planning and improving at test-time via refinement. Our model demonstrably improves over the sampling-based planning in the real world, operating 2.5x-20x faster than AIT* while improving by over 20% in terms of motion planning success rate. Notably, our model generalizes to a wide distribution of task instances and demonstrates favorable scaling properties. At the same time, there is significant room for future work to improve upon, our model 1) is susceptible to point-cloud quality, which may require improving 3D representations via implicit models such as NeRFs [72], 2) does not still handle tight spaces well, a capability which could be potentially acquired via RL fine-tuning of the base policy.

Chapter 4

DRP: Moving Beyond Data 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* [38] and AIT* [102], 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 [6, 53, 86, 107], such as Riemannian Motion Policies (RMP) [86] and Geometric Fabrics [107], offer reactive collision avoidance in dynamic scenes. However, these approaches lack global scene awareness and often become trapped in local minima in complex environments [32].

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 [24, 32, 33, 82]. Unfortunately, prior methods such as MπNet [32] 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 [24], 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 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 [104], 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

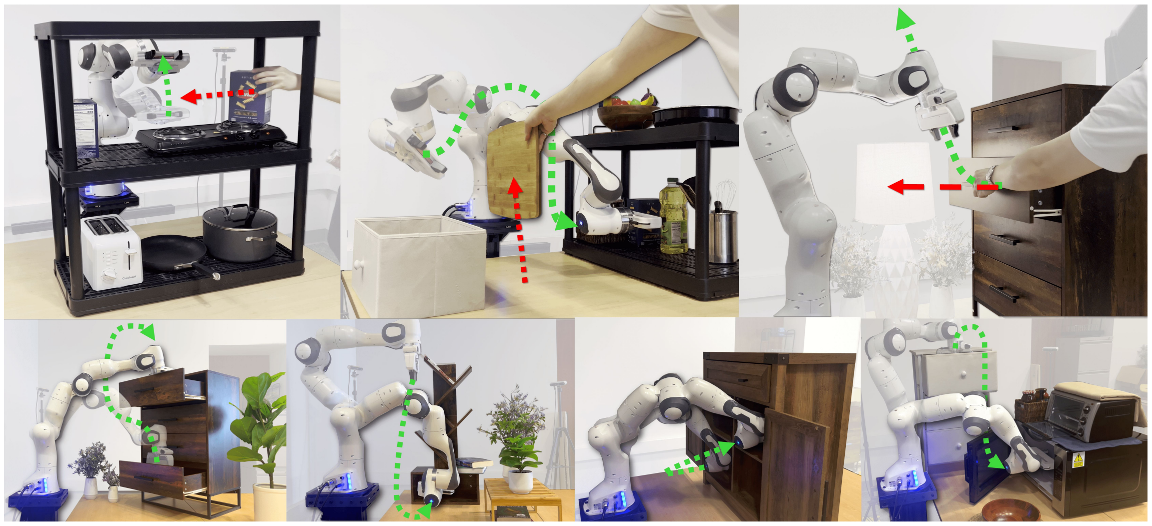

Figure 4.1: We present Deep Reactive Policy (DRP), a point cloud conditioned motion policy capable of performing reactive, collision-free goal reaching in diverse complex and dynamic environments.

obstacle arrangements and dynamic obstacles. DRP consistently outperforms previous state-of-the-art motion planning methods, as detailed in Section 4.3. Video results available at deep-reactive-policy.github.io.

4.1 Related Work

4.1.1 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* [38] and its variants [56, 64, 65] guarantee optimality under admissible heuristics but scale poorly in high-dimensional continuous domains due to discretization. Sampling-based planners such as PRM [52], RRT [60], and their extensions [7, 36, 57, 61, 102] improve scalability and efficiency through continuous-space sampling and informed exploration. Trajectory optimization methods [30, 95, 119] 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.

4.1.2 Locally-Reactive Methods

Locally reactive controllers generate collision-free motion by steering toward goals while avoiding nearby obstacles [6, 53, 86, 107]. Though effective in dynamic settings, they often get trapped in local minima within cluttered environments [32]. 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.

4.1.3 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 [16, 59, 82, 116], guide search [43, 84] or initialize optimization [40]. 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 [15, 46, 94] 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 [32] and MπFormer [33] use supervised learning on point cloud inputs, achieving strong in-distribution results, but struggling to generalize due to limited training diversity. NeuralMP [24] 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 Method

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 [104], 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 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 [107], 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 [24], 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

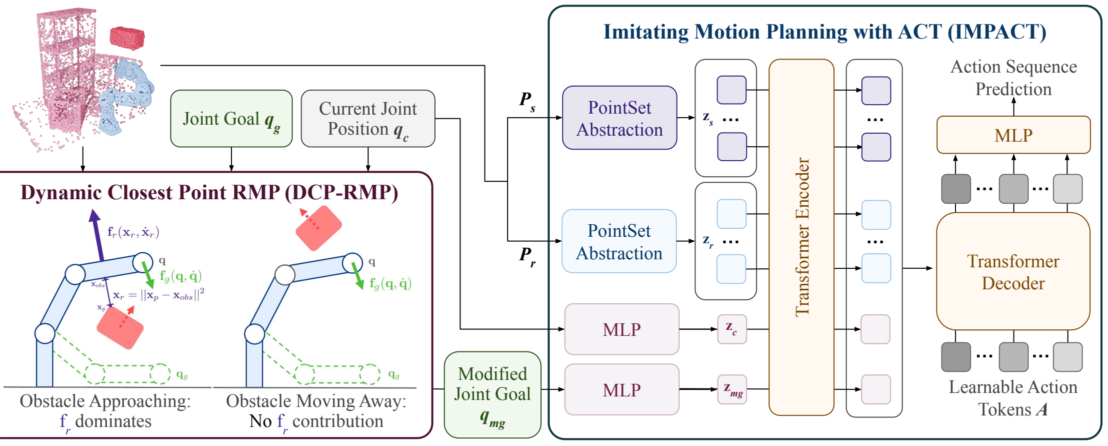
Imitating Motion Planning with ACT (IMPACT)
Joint Goal q_g
Current Joint Position q_c
Dynamic Closest Point RMP (DCP-RMP)
Obstacle Approaching: f_r dominates
Obstacle Moving Away: No f_r contribution
Modified Joint Goal q_{mg}
PointSet Abstraction
MLP
Transformer Encoder
Transformer Decoder
Action Sequence Prediction
Learnable Action Tokens A

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.

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 $q_{mg} \in \mathbb{R}^7$.

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++ [79] 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 $[q_1, q_2, \ldots, q_S]$:

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

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

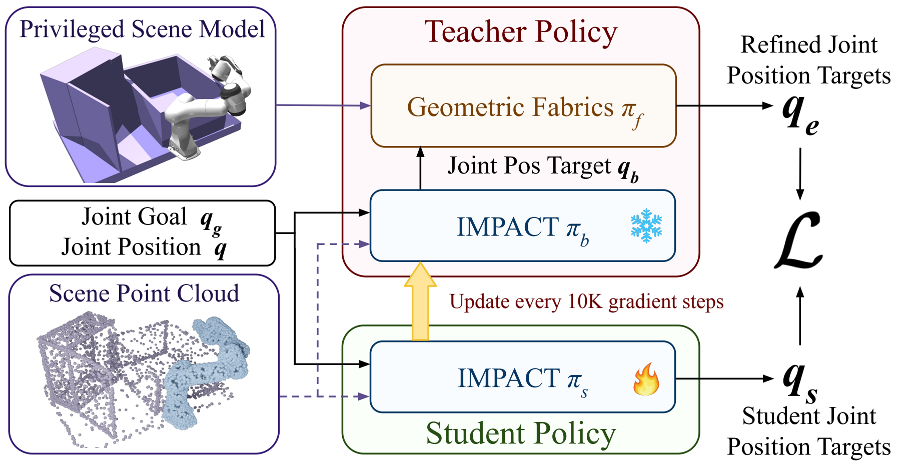

Figure 4.3: The IMPACT policy is combined with a locally reactive Geometric Fabrics controller to enable improved obstacle avoidance. This combined teacher policy is then distilled into a point-cloud conditioned student policy.

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 [107], 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 [92].

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 [68]. 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 10000 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

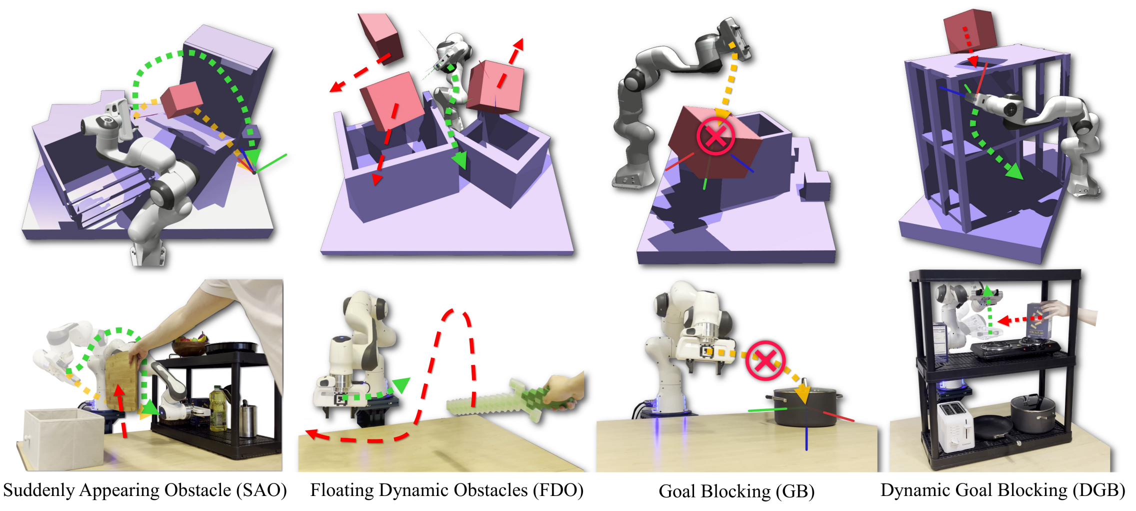
Suddenly Appearing Obstacle (SAO)
Floating Dynamic Obstacles (FDO)
Goal Blocking (GB)
Dynamic Goal Blocking (DGB)

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.

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 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

	DRPBench					MπNets Dataset
	SE	SAO	FDO	GB	DGB	
With privileged information:						
AIT* [102]	40.50	0	0	0	0	89.22
cuRobo [104]	82.97	59.00	39.50	0	3.00	**99.78**
cuRobo-Vox [104]	50.53	58.67	40.50	0	2.50	-
RMP [86]	32.97	46.0	49.50	**71.08**	50.50	41.90
MπNets [32]	2.50	0.33	0	0	0	65.18
MπFormer [33]	0.19	0	0	0	0	30.02
NeuralMP [24]	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 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.

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 Experiment Setup and Results

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 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 [32] 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 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

	FDO		DGB	
Has DCP-RMP?	✗	✓	✗	✓
cuRobo [104]	39.50	51.50	3.00	54.50
NeuralMP [24]	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 [24]	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.

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 Conclusion

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.

4.5 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 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

Conclusions

This thesis presents a data-driven framework for learning generalizable robotic policies at scale. We systematically explore how to collect diverse demonstrations, leverage them for training, and design policies that operate robustly across tasks and settings.

Through BiDex, we enable scalable collection of real-world demonstrations with a low-cost and portable teleoperation setup. These demonstrations allow us to train visuomotor policies for complex bimanual manipulation. We then show that large-scale simulated motion planning data can be distilled into compact neural policies (Neural MP) that generalize to unseen environments.

To address the limitations of naive imitation learning, we propose DRP, which leverage online finetuning and additional reactive control module to achieve stronger performance and generalization.

Across these components, we demonstrate that data-driven learning can enable dexterous and generalizable robot policies. Looking forward, promising directions include integrating world models to enable long-horizon reasoning and task generalization, and augmenting imitation learning with online reinforcement learning for continual adaptation and improved robustness. Ultimately, we hope this work contributes to a broader vision of robots that learn and generalize like humans—through experience, feedback, and large-scale interaction with the world.

Bibliography

[1] Ananye Agarwal, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Legged locomotion in challenging terrains using egocentric vision. In *Conference on robot learning*, pages 403–415. PMLR, 2023. 3, 3.1

[2] Ananye Agarwal, Shagun Uppal, Kenneth Shaw, and Deepak Pathak. Dexterous functional grasping. In *Conference on Robot Learning*, pages 3453–3467. PMLR, 2023. 2.3.1

[3] AgileX Robotics. Ranger mini. `https://global.agilex.ai/products/ranger-mini`, 2023. Omnidirectional mobile robot platform. 2.2.3

[4] 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. 2

[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. 2.1

[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. 4, 4.1.2

[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. 4.1.1

[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. 2.2.3

[9] Gerald Brantner and Oussama Khatib. Controlling ocean one: Human–robot collaboration for deep-sea manipulation. *Journal of Field Robotics*, 38(1):28–51, 2021. 2.1

[10] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022. 3.1

[11] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023. 3.1

[12] Samuel R Buss and Jin-Su Kim. Selectively damped least squares for inverse kinematics. *Journal of Graphics tools*, 10(3):37–49, 2005. 2.2.1

[13] 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. 2.4.2

[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. 3.1

[15] 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. 4.1.3

[16] 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. 4.1.3

[17] Constantinos Chamzas, Carlos Quintero-Pena, Zachary Kingston, Andreas Orthey, Daniel Rakita, Michael Gleicher, Marc Toussaint, and Lydia E Kavraki. Motionbenchmaker: A tool to generate and benchmark motion planning datasets. *IEEE Robotics and Automation Letters*, 7(2):882–889, 2021. 3.1

[18] Tao Chen, Megha Tippur, Siyang Wu, Vikash Kumar, Edward Adelson, and Pulkit Agrawal. Visual dexterity: In-hand reorientation of novel and complex object shapes. *Science Robotics*, 8(84):eadc9244, 2023. 3.2.2

[19] Xuxin Cheng, Kexin Shi, Ananye Agarwal, and Deepak Pathak. Extreme

parkour with legged robots. *arXiv preprint arXiv:2309.14341*, 2023. 3, 3.1

[20] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023. 2.1

[21] 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. 2.1

[22] Sammy Christen, Wei Yang, Claudia Pérez-D'Arpino, Otmar Hilliges, Dieter Fox, and Yu-Wei Chao. Learning human-to-robot handovers from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3.2.2

[23] Murtaza Dalal, Ajay Mandlekar, Caelan Garrett, Ankur Handa, Ruslan Salakhutdinov, and Dieter Fox. Imitating task and motion planning with visuomotor transformers. 2023. 3.1, 3.2.2

[24] 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. 4, 4.1.3, 4.2.1, **??**, **??**, **??**

[25] 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. *arXiv preprint arXiv:1910.11215*, 2019. 2.1

[26] 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. 2.4.2

[27] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador, Winson Han, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. Procthor: Large-scale embodied ai using procedural generation. *Advances in Neural Information Processing Systems*, 35:5982–5994, 2022. 3.1

[28] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023. 3.1, 3.2.1

[29] Runyu Ding, Yuzhe Qin, Jiyue Zhu, Chengzhe Jia, Shiqi Yang, Ruihan Yang, Xiaojuan Qi, and Xiaolong Wang. Bunny-visionpro: Bimanual dexterous teleoperation with real-time retargeting using vision pro. 2024. 2, 2.1, 2.3.1

[30] 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. 3, 4.1.1

[31] 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. 3.1, 3.2.2, **??**, **??**, **??**, 3.4

[32] 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. 4, 4.1.2, 4.1.3, **??**, 4.3.1

[33] 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. 4, 4.1.3, **??**

[34] 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. 2.1

[35] Zipeng Fu, Tony Z Zhao, and Chelsea Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024. 2.1

[36] 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. doi: 10.1109/ICRA.2015.7139620. 4.1.1

[37] 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. 2.1, 2.2.1

[38] 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. 4, 4.1.1

[39] 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. 3

[40] Huang Huang, Balakumar Sundaralingam, Arsalan Mousavian, Adithyavaira-

van Murali, Ken Goldberg, and Dieter Fox. Diffusionseeder: Seeding motion optimization with diffusion for rapid motion planning. *arXiv preprint arXiv:2410.16727*, 2024. 4.1.3

[41] Siyuan Huang, Zan Wang, Puhao Li, Baoxiong Jia, Tengyu Liu, Yixin Zhu, Wei Liang, and Song-Chun Zhu. Diffusion-based generation, optimization, and planning in 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16750–16761, 2023. 3.1

[42] 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. 2

[43] Brian Ichter, Pierre Sermanet, and Corey Lynch. Broadly-exploring, local-policy trees for long-horizon task planning. *arXiv preprint arXiv:2010.06491*, 2020. 4.1.3

[44] Brian Ichter, Pierre Sermanet, and Corey Lynch. Broadly-exploring, local-policy trees for long-horizon task planning. *arXiv preprint arXiv:2010.06491*, 2020. 3.1

[45] Yunfan Jiang, Chen Wang, Ruohan Zhang, Jiajun Wu, and Li Fei-Fei. Transic: Sim-to-real policy transfer by learning from online correction. *arXiv preprint arXiv: Arxiv-2405.10315*, 2024. 3.2.2

[46] 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. 4.1.3

[47] 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. 3.1

[48] 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. 2

[49] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011. 3

[50] Pushkal Katara, Zhou Xian, and Katerina Fragkiadaki. Gen2sim: Scaling up robot learning in simulation with generative models, 2023. 3.1

[51] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Proba-

bilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996. 3

[52] 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. 4.1.1

[53] 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. 4, 4.1.2

[54] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90–98, 1986. 3

[55] 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. 2.1, 3, 3.1

[56] Sven Koenig and Maxim Likhachev. A new principle for incremental heuristic search: Theoretical results. In *ICAPS*, pages 402–405, 2006. 3, 4.1.1

[57] 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. 3, 4.1.1

[58] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021. 2

[59] 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. 4.1.3

[60] Steven LaValle. Rapidly-exploring random trees: A new tool for path planning. *Research Report 9811*, 1998. 4.1.1

[61] 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. 3, 4.1.1

[62] 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. 2

[63] 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. doi: 10.1109/COASE.2017.8256067. 2.1

[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. 3, 4.1.1

[65] 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. 4.1.1

[66] 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. 2.1, 2.2.1

[67] 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. 2.1

[68] 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. 4.2.2

[69] 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. *arXiv preprint arXiv:2108.03298*, 2021. 2.1

[70] 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. doi: 10.1109/Humanoids57100.2023.10375195. 2, 2.3.1

[71] Manus. `https://www.manus-meta.com/`. [VR haptic gloves]. 2, 2.2, 2.2.1

[72] 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. *Communications of the ACM*, 65(1):99–106, 2021. 3.5

[73] Movella. `https://www.movella.com/products/xsens#overview`. [Motion capture technology]. 2.1

[74] Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv*

preprint arXiv:2310.08864, 2023. 2.1, 3, 3.1

[75] Younghyo Park and Pulkit Agrawal. Using apple vision pro to train and control robots, 2024. URL `https://github.com/Improbable-AI/VisionProTeleop`. 2, 2.1, 2.3.1

[76] Georgios Pavlakos, Dandan Shan, Ilija Radosavovic, Angjoo Kanazawa, David Fouhey, and Jitendra Malik. Reconstructing hands in 3D with transformers. In *CVPR*, 2024. 2.2.1, 2.3.1

[77] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022. 3.1

[78] Psyonic. `https://www.psyonic.io`. [Bionic hand]. 2.1

[79] 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. 4.2.1

[80] 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. 3.2.2

[81] 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. 2.1

[82] 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. 4, 4.1.3

[83] 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. 3.1, **??**, 3.4

[84] 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. 4.1.3

[85] 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. 3.1

[86] Nathan D Ratliff, Jan Issac, Daniel Kappler, Stan Birchfield, and Dieter Fox. Riemannian motion policies. *arXiv preprint arXiv:1801.02854*, 2018. 4, 4.1.2, **??**

[87] 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. 2.1

[88] Daniel Ritchie, Kai Wang, and Yu-an Lin. Fast and flexible indoor scene synthesis via deep convolutional generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6182–6190, 2019. 3.1

[89] Rokoko. `https://www.rokoko.com/`. [Motion capture hardware and software]. 2.1, 2.2.1

[90] Javier Romero, Dimitrios Tzionas, and Michael J Black. Embodied hands: Modeling and capturing hands and bodies together. *arXiv preprint arXiv:2201.02610*, 2022. 2.1

[91] 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. 2.1, 2.2.1

[92] 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. 4.2.2

[93] 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. *arXiv preprint arXiv:2309.11414*, 2023. 3.1, 3.2.2, **??**, 3.4

[94] 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. 4.1.3

[95] 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. 3, 4.1.1

[96] Shadow Robot Company. `https://www.shadowrobot.com/`. [Robotic hand]. 2.1, 2.3.1

[97] 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. URL `https://openreview.net/forum?id=eQomRzRZEP`. 2.2

[98] Kenneth Shaw, Ananye Agarwal, and Deepak Pathak. Leap hand: Low-

cost, efficient, and anthropomorphic hand for robot learning. *arXiv preprint arXiv:2309.06440*, 2023. (document), 2, 2.1, 2.2, 2.2, 2.3.2, 2.4, 2.2

[99] 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. doi: 10.1177/02783649241227559. URL https://doi.org/10.1177/02783649241227559. 2

[100] Aravind Sivakumar, Kenneth Shaw, and Deepak Pathak. Robotic telekinesis: Learning a robotic hand imitator by watching humans on youtube. *arXiv preprint arXiv:2202.10448*, 2022. 2.1, 2.2.1

[101] 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. 2.1

[102] 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. 4, 4.1.1, ??

[103] 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. 3, 3.2.1, ??, ??

[104] 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. 4, 4.2, ??, ??, ??

[105] 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. 3, 3.3, ??

[106] Valve Corporation. https://store.steampowered.com/steamvr. [Virtual reality platform]. 2, 2.1

[107] 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. 4, 4.1.2, 4.2, 4.2.2

[108] Vicon. https://www.vicon.com/. [Motion capture technology]. 2

[109] 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. 2.1

[110] 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. *arXiv preprint arXiv:2403.07788*, 2024. 2, 2.1, 2.2.1, 2.2.3

[111] Xinpeng Wang, Chandan Yeshwanth, and Matthias Nießner. Sceneformer: Indoor scene generation with transformers. In *2021 International Conference on 3D Vision (3DV)*, pages 106–115. IEEE, 2021. 3.1

[112] Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan. Robogen: Towards unleashing infinite data for automated robot learning via generative simulation. *arXiv preprint arXiv:2311.01455*, 2023. 3.1

[113] Charles W Warren. Global path planning using artificial potential fields. In *1989 IEEE International Conference on Robotics and Automation*, pages 316–317. IEEE Computer Society, 1989. 3

[114] 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. 2, 2.1, 2.2, 2.2.2

[115] 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. 2.2.3

[116] 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. 4.1.3

[117] 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. (document), 2, 2.2.2, 2.2.3, 2.3, 2.4.2

[118] 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. 2.1, 3.2.2

[119] 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. 4.1.1